НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

До захисту допущено:

В.о. завідувача кафедри

_____ Оксана ТИМОЩУК

«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системний аналіз і управління»

спеціальності 124 «Системний аналіз»

на тему: « Розпізнавання мови на основі апарату штучних нейронних мереж»

Виконав (-ла):
студент (-ка) IV курсу, групи КА-63
Ільченко Юлія Едуардівна _____

Керівник:
професор каф. ММСА, д.т.н., проф.
Мухін Вадим Євгенович _____

Консультант з економічного розділу:
доцент, к. е. н. Шевчук О. А. _____

Рецензент:
доцент, к.т.н., доцент кафедри технічної кібернетики
Корнага Ярослав Ігорович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 124 «Системний аналіз»

Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Оксана ТИМОЩУК
«___»_____20__ р.

ЗАВДАННЯ
на дипломну роботу студенту
Ільченко Юлії Едуардівні

1. Тема роботи «Розпізнавання мови на основі апарату штучних нейронних мереж», керівник роботи Мухін Вадим Євгенович, д. т. н, професор, затверджені наказом по університету від «_25_»_травня_ 20_20_ р. №_1143-с__

2. Термін подання студентом роботи 08 травня 2020 року

3. Вихідні дані до роботи : словник команд, мовна модель, акустична модель.

4. Зміст роботи : дослідження предметної області систем розпізнавання, процес розпізнавання мовлення на основі апарату штучних нейронних мереж.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) : презентація.

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Шевчук О. А., доцент		

7. Дата видачі завдання 13 квітня 2020 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Затвердження теми БДР	13.04.2020-19.04.2020	Виконано
2.	Ознайомлення зі структурою БДР згідно з Положенням про державну атестацію студентів НТУУ «КПІ»	20.04.2020-26.04.2020	Виконано
3.	Ознайомлення з ДСТУ 3008-2015 та стандарти ЄСПД	27.04.2020-03.05.2020	Виконано
4.	Проведення дослідження за темою БДР під керівництвом керівника	04.05.2020-10.05.2020	Виконано
5.	Завершення роботи над першим варіантом частини БДР	11.05.2020-17.05.2020	Виконано
6.	Проведення роботи над експериментальною частиною БДР	18.05.2020-24.05.2020	Виконано
7.	Проведення роботи над програмним продуктом	25.05.2020-31.05.2020	Виконано
8.	Оформлення БДР та аналіз отриманих результатів	01.06.2020-07.06.2020	Виконано

Студент Ю. Е. Ільченко

Керівник В. Є. Мухін

РЕФЕРАТ

Дипломна робота містить 88 с., 5 ч., 20 рис., 7 табл., 2 дод., 19 джерел.

РОЗПІЗНАВАННЯ МОВИ, НЕЙРОННІ МЕРЕЖІ, СИНТЕЗАТОРИ МОВЛЕННЯ

Об’єкт дослідження – процес розпізнавання мови на основі апарату штучних нейронних мереж.

Мета роботи – дослідити існуючі системи розпізнавання, проаналізувати методи та алгоритми розпізнавання мови, розробка системи голосового локального управління комп’ютером.

Методи дослідження – аналіз процесу розпізнавання мови, створення програмного продукту, експериментальне дослідження отриманих результатів.

Результатом роботи є система розпізнавання “Jarvis”, а саме система голосового локального управління комп’ютером, яка виступає в ролі персонального помічника, виконує команди та тим самим пришвидшує роботу користувача. Розроблений додаток може бути додано як файл автозавантаження, що надасть змогу запускати програму одночасно з операційною системою.

Система розпізнавання “Jarvis” є інтерактивною програмою і може виконувати команди різного характеру від пошуку прогнозу погоди до дзвінка у мережі «Skype».

ABSTRACT

This thesis includes 88 p., 5 sections, 20 pict., 7 tabl., 2 appendixes, 19 sources.

SPEECH RECOGNITION, NEURAL NETWORKS, SPEECH SYNTHESIZERS

The object of research is the process of speech recognition based on the apparatus of artificial neural networks.

The purpose of the work is to investigate the existing recognition systems, to analyze the methods and algorithms of speech recognition, to develop a system of voice local control of the computer.

Research methods - analysis of the process of language recognition, software product development, experimental research of the results.

The result is a recognition system “Jarvis”, a voice-based local computer control system that acts as a personal assistant, executes commands, and thus speeds up the user's work. The developed application can be added as a autoload file, which will allow you to run the program at the same time as the operating system.

The recognition system “Jarvis” is an interactive program and can execute different commands, from searching for the weather forecast to a call in the “Skype” application.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

AWS – Amazon Web Services

API – Application Programming Interface

MFCC – Mel-frequency cepstrum coefficients

LPC – Linear Preedictive Coding

АЧХ – Амплитудно-частотная характеристика

DCT – Discrete Cosine Transform

DTW – Dynamic Time Wrapping

DTW – Dynamic Time Wrapping

ПММ – Приховані Марківські моделі

NN – Neural Networks

IDE – Integrated Development Environment

CDDL – Common Development and Distribution License Computing

ВСТУП

Люди навчились промовляти слова задовго до того, як писати їх. Ми можемо говорити по 150 слів за хвилину, в свою чергу за ті самі 60 секунд середньостатистична людина може написати всього на всього 40 слів. Поки клавіатура і миша залишаються найбільш поширеними пристроями для взаємодії з комп'ютером. Однак людям набагато звичніше передавати інформацію за допомогою мови, жестів, міміки. Спілкування з технологічними пристроями за допомогою голоса стає з кожним днем популярнішим і природним: ідучи по вулиці або сидячи в транспорті, завжди зручніше задати той чи інший запит в пошукову систему голосом, а не друкуючи. Системи розпізнавання мови стали невід'ємною частиною нашого життя, вони надають можливість комп'ютерам та програмам слухати мовлення користувача та перетворювати це на текст, що приносить збільшення ефективності роботи в свою чергу.

Перетворення розмови на текст ніколи не є 100% правильним. Ця ідея є досить незвичною для розробників програмного забезпечення, які зазвичай працюють з детермінованими системами. І це створює безліч питань, характерних лише для мовленнєвих технологій. На сьогоднішній день розпізнавання мови є однією з найскладніших задач. Ми все ще перебуваємо на деякому віддаленні від реалізації справжнього потенціалу технології розпізнавання голосу. Проблема стосується як витонченості самої технології, так і її інтеграції в наше життя. Поточні цифрові асистенти можуть дуже добре інтерпретувати мову, але вони не є діалоговими інтерфейсами, яких очікують технологічні постачальники. Більш того, розпізнавання мови так і залишається обмеженим не дуже невеликою кількістю готових продуктів.

Тому об'єктом даної роботи стало створення системи розпізнавання з діалоговим інтерфейсом, яка буде виступати в ролі персонального

помічника та виконувати дії задані голосом, що значно пришвидшує і спрощує роботу людини.

У першій главі буде освітлено історичне підгрунтя для розвитку систем, а також наведено порівняльний аналіз існуючих методів розпізнавання людини по голосу. У главі 2 розглядається структура мовлення, наводиться опис властивостей і параметрів голосу та загальні поняття про нейронні мережі. З основних методів для розпізнавання мови буде обрано найбільш підходящі для виконання нашої задачі. Глава 3 присвячена опису нашої програми реалізації розпізнавання мови. Буде наданий опис експериментальних досліджень та аналіз отриманих результатів у главі 4. В останній главі буде проведений функціонально-вартісний аналіз роботи.

Результатом роботи є система розпізнавання “Jarvis”, а саме система голосового локального управління комп’ютером, яка виступає в ролі персонального помічника, виконує команди та тим самим пришвидшує роботу користувача.

Глава 1. Історія та класифікація систем. Порівняльний аналіз існуючих методів розпізнавання людини по голосу

1.1 Історія виникнення та розвитку систем розпізнавання мови

Технологія розпізнавання мови увійшла в суспільство порівняно недавно, з блискучими подіями запуску від високотехнологічних гігантів провідних світових трендів. Історія виникнення систем розпізнавання мови дуже багата і різноманітна, яка починалась ще визначенням окремих слів, надалі ще більших словників і нарешті до швидких відповідей на запитання, як це робить наприклад Siri.

Перший засіб для розпізнавання мови з’явився ще в далекому 1952 році. Bell Laboratories розробили систему "Audrey", яка розпізнавала вимовлені людиною цифри. Враховуючи складність мови, це вірно, що інженери спочатку фокусуватись на цифрах. Та вже через десять років в 1962 році IBM випустила їх розроблену систему "Shoebox", яка розуміла 16 слів англійською.

Лабораторії в США, Японії, Англії та СРСР розробили ще кілька апаратів, які розпізнавали окремі вимовлені звуки, розширивши технологію розпізнавання мови підтримкою чотирьох голосних і дев'яти приголосних звуків. Звучали вони не дуже добре, але ці перші спроби дали вражаючий старт, особливо якщо враховувати, наскільки примітивними були комп'ютери того часу.

Завдяки новим підходам і технологіям словниковий запас подібних систем виріс з декількох сотень до декількох тисяч слів і мав потенціал розпізнавання необмеженої кількості слів. Однією з причин був новий статистичний метод, більше відомий як прихована марківських модель. Використовуючи шаблони для слів і звукові патерни, вона розглядала ймовірність того, що невідомі звуки могли бути словами. Ця база використовувалася іншими системами ще протягом двадцяти років.

Тільки в 1997 році випустили перший у світі «безперервний розпізнавач мови», тобто більше не доводилося робити паузу між кожним словом, у вигляді програмного забезпечення Dragon's NaturallySpeaking. Цей винахід був здатний розуміти 100 слів за хвилину, він все ще використовується сьогодні в оновленій формі і користується попитом у лікарів.

Проривом для технології розпізнавання мови стало машинне навчання, завдяки чому кількість помилок при розпізнаванні слів стала значно меншою. Google об'єднав новітні технології з хмарними обчисленнями для обміну даними і це призвело до запуску додатка Google Voice Search для iPhone у 2008 році. Google ввів елементи персоналізації в свої результати пошуку за допомогою голосу, і використовував ці дані для розробки свого алгоритму Hummingbird, отримуючи набагато більше тонке розуміння використовуваної мови.

Потім з'явилася Siri, яка так само, як і система Google Voice Search, покладається на хмарні обчислення. Siri використовує ті дані, які їй відомі про тебе, щоб згенерувати відповідь, яка випливає з контексту і відповідає на твій запит як деяка особистість. Розпізнавання мови перетворилося з інструмента у розвагу.

Методи, які використовувались для здійснення цих проривів вперед, стали витонченішими в тій мірі, в якій вони тепер вільно симулюють принципи, засновані на схемах роботи людського мозку. Комп'ютери на базі хмарних обчислень увійшли в мільйони будинків і можуть контролюватися голосом, навіть пропонуючи інтерактивні відповіді на широкий спектр запитів.

На сьогоднішній день технологія дійсно корисна для виконання повсякденних завдань. Запитайте у Siri, Cortana або Google, яка погода буде завтра, і вона надасть цілком виразний словесний звіт. Звичайно пристрій ще не повністю досконалий, але розпізнавання мови досягло зараз

прийнятного рівня точності для більшості людей, причому всі основні платформи повідомляють про частоту помилок менше 5%.

1.1 Класифікація систем розпізнавання мови

Нижче на рисунку 1.1 наведена класифікація систем розпізнавання мови.

Рисунок 1.1 – Класифікація систем розпізнавання мови

1.3.1 Типи систем розпізнавання мови за послідовністю слів

Усі сучасні описи мовлення певною мірою є імовірнісними. Це означає, що між одиницями чи між словами немає певних пауз. В системах розпізнавання мови можна умовно виділити декілька класів, які послідовності слів вони мають можливість аналізувати:

1. Окремі слова

Такі системи розраховані на то, що вимова кожного слова буже оточено тишою з обох сторін, тобто до і після слова повинна слідувати пауза. Зручно використовувати у ситуаціях, коли потрібно сказати якусь команду, яка складається з одного слова. В такій системі окремою задачою виявляється виділення пауз у мові між сусідіми входженнями, однак таку задачу все ж таки найлегше реалізувати навідміну від наступних.

2. Зв’язна мова

Ці системи подібні до попередніх, але деяким висловлюванням дозволяється іти один за одним з мінімальною паузою. Тут під висловленням мається на увазі слова, які представляють єдине значення до комп’ютора.

3. Неперервна мова

Близька до природньої мови, тобто користувачам дозволяється вимовляти слова без паузи. Таку систему достатньо важко розробити.

4. Суцільна мова (спонтанна мова)

Такі системи здатні розпізнавати природню мову людини. На сьогодні це одна з найскладніх задач.

1.3.2 Типи систем розпізнавання мови за розміром словника

Розмір лексики має важливе значення для системи розпізнавання мови та її роботи. Словниковий запас визначається як набір слів, з яких система може вибрати, тобто слова, на які він може посилатися. У випадках, коли варіантів небагато, розпізнавання очевидно простіше, ніж якщо словниковий запас великий. Так виділяють такі типи на основі об'єму словника:

- Малий словник (розпізнає від 1 до 100 слів або фраз)
- Середній словник (розпізнає від 101 до 1000 слів або фраз)
- Великий словник (розпізнає від 1001 до 10000 слів або фраз)
- Дуже великий словник (розпізнає більше ніж 10 000 слів або фраз)

1.3.3 Типи систем розпізнавання мови по залежності від диктора

Кожна людина має свої особливості та властивості голосу, які будуть розглянуті далі в цій главі. Базуючись на цьому можна виокремити дві типи систем:

Дикторозалежні моделі

Програмне забезпечення для розпізнавання мови, яке залежить від знання особливостей голосу мовця. Програмне забезпечення вивчає характеристики голосу мовця через голосове навчання. Цей тип системи повинен бути навчений конкретному користувачеві, перш ніж він зможе розпізнати сказане. Цей тип системи працює добре, якщо буде лише один користувач, який буде говорити з системою.

Системи, що залежать від ораторів, як правило, здатні розпізнавати мовлення з різних контекстів (слів, фраз). Однак незалежні від оратора системи здатні розпізнавати мовлення у різних користувачів, обмежуючи контексти мови (слова та фрази).

Дикторонезалежні моделі

Програмне забезпечення, яке не потребує навчання, не залежить від ораторів. Ці системи використовуються для автоматизованих телефонних інтерфейсів. Ці системи можуть бути використані різними особами без навчання розпізнаванню мовленнєвих особливостей кожної людини.

1.2 Порівняння існуючих систем

Розглянемо найпопулярніші постачальники послуг з розпізнавання голосу на ринку.

Dragon Naturally Speaking (його ще називають Dragon for PC) є найкращим загальним програмним забезпеченням для розпізнавання голосу. Його можна використовувати як в особистих, так і в офіційних

цілях. Так Dragon Home допоможе вам у кількох повсякденних заходах, таких як диктування домашніх завдань, надсилання електронної пошти та навіть у веб-серфінгу. Dragon Professional Individual допомагає працюючим особам та малому бізнесу створювати та переписувати документи, вставляючи підпис або налаштовуючи словниковий запас. Dragon Legal Individual - це допомога в юридичних професіональних справах таких, як впорядкування юридичної документації.

Google Cloud Speech API є лідером індустрії. Це програмне забезпечення розпізнає 120 мов, яке навчилось розпізнавати різних спікерів і самостійно визначати мову записи з декількох можливих. Google Cloud Speech API розшифровує аудіо з кол-центрів, перекладає, підтримує глобальних користувачів, самостійно розставляє знаки пунктуації та виконує команди. Він навіть використовує власний "мозок" для транскрипції попередньо записаного звуку. Це дивовижний інструмент. Також він підтримується всією підтримкою, якою Google так відомий.

Amazon Lex надає інструменти для вирішення складних завдань глибокого навчання, таких як розпізнавання мови і розуміння мови, за допомогою простого і повністю керованого сервісу. Amazon Lex інтегрований з сервісом AWS Lambda, який можна використовувати для простого доступу до більшості функцій і виконання функціонального коду на сервері з метою вилучення та оновлення даних. Після створення бот можна розгортати безпосередньо на платформі чату, мобільному клієнті або пристрої IoT. Можна також використовувати надані звіти для відстеження метрик свого бота. Amazon Lex надає безпечне і просте у використанні комплексне рішення, що дозволяє створювати, публікувати боти і відстежувати їх стан.

Усім відомий Siri- це найпопулярніший особистий помічник в Америці. Siri може дзвонити і відправляти повідомлення, коли ви за кермом, у вас зайняті руки або ви просто кудись квапитесь. Siri вгадає то,

що може знадобитися, наприклад, запропонує відправити повідомлення колегам, якщо ви спізнюєтеся на зустріч. Siri допоможе працювати з пристроєм, не торкаючись до нього. Розумний помічник запам'ятовує ваші звички і передбачає, що вам може знадобитися протягом дня. Ви можете перевіряти факти, робити розрахунки або переводити фрази з однієї мови на іншу. З 95-відсотковою точністю Сірі перевершує всіх своїх гігантів з Кремнієвої долини. Точність та інтелект помічника і надалі будуть продовжувати вдосконалюватися.

Наступний лідер ринку Voice Finger використовується в цілях схожих до нашої розробленої програми. За допомогою Voice Finger ви зможете керувати комп'ютером лише голосом і не потрібно буде використовувати клавіатуру та мишу. Він підтримує команди Windows. За допомогою цього інструменту ви зможете виконувати завдання з нульовим контактом на комп’ютері.

1.3 Синтезатори мовлення

У той час як завдання розпізнавання мови дуже складна і вирішена лише частково, завдання синтезу мови набагато простіше (хоча і там є чимало проблем, що чекають свого рішення).

У побуті нам доводиться стикатися з різними системами синтезу мови. Ось кілька прикладів.

Технології синтезу мови застосовуються в метро при оголошенні зупинок.

Власники мобільних телефонів можуть спілкуватися з автоматичною сервісною службою для визначення залишку коштів на рахунку, перемикання тарифних планів, підключення або відключення послуг тощо. Сервісна служба спілкується голосом із застосуванням технологій синтезу мови.

Випущено чимало дитячих іграшок, що «говорять» людським голосом. У цих іграшках також застосовуються найпростіші синтезатори мови або цифрові магнітофони.

Синтезатори мови застосовуються в різних голосових системах попередження, що встановлюються в автомобілях і літаках. Такі системи дозволяють привернути увагу людини до виникнення тієї чи іншої критичної ситуації, не відволікаючи його від процесу керування автомобілем, літаком чи іншим аналогічним засобом.

Також розроблено чимало комп'ютерних програм, здатних читати голосом вміст текстових файлів або текст, розташований на панелі програм. Ці системи можуть виявитися корисними тим, у кого ослаблений або повністю відсутній зір.

1.5.1 Моделі синтезу мови

Всі існуючі в даний час методи синтезу людської мови засновані на використанні двох моделей - моделі компілятивного синтезу і формантно-голосові моделі, як показано на рисунку 1.2.

Рисунок 1.2 – Моделі синтезаторів мовлення

Розглянемо детальніше кожну з цих моделей нижче.

Модель компілятивного синтезу передбачає синтез мови шляхом конкатенації (складання) записаних зразків окремих звуків, вимовлених диктором.

При використанні цієї моделі складається база даних звукових фрагментів, з яких в подальшому буде синтезуватися мова.

На перший погляд цей підхід не повинен викликати особливих труднощів.

Модель компілятивного синтезу підходить, головним чином, тільки в найпростіших випадках, коли синтезатор повинен вимовляти відносно невеликий і заздалегідь відомий набір фраз. При цьому забезпечується досить висока якість мови. Втім, цей факт не дуже дивний, якщо згадати, що для синтезу використовується природна людська мова.

Проте, на стику складених звукових фрагментів можливі інтонаційні викривлення і розриви, помітні на слух. Крім того, створення великої бази даних звукових фрагментів, що враховує всі особливості вимови фонем і алофонів з різними інтонаціями, являє собою складну і копітку роботу.

Формантно-голосова модель заснована на моделюванні мовного тракту людини. Ця модель може бути реалізована із застосуванням нейронних мереж і допускає самонавчання. На жаль, зважаючи на складність точного моделювання особливостей мовного тракту, а також з врахуванням інтонаційної модуляції мови формантно-голосова модель володіє відносно низькою точністю синтезованих звуків мови. Проте, сучасні програми синтезу мови, побудовані з використанням цієї моделі, синтезують цілком розбірливу мову і можуть застосовуватися в ряді випадків.

Треба зауважити, що системи голосового попередження про виникнення аварійних ситуацій краще будувати з використанням моделі компілятивного синтезу, так як розбірливість мови в таких системах виходить на передній план.

Що ж стосується «побутових» синтезаторів мови, то в них можна з успіхом застосовувати і формантно-голосову модель.

1.5.2 Порівняльний аналіз синтезаторів мовлення

Нижче наведено інформацію про декілька програмних реалізацій синтезаторів мови. Більшість таких синтезаторів розроблено для платформи Microsoft Windows і користується мовним програмним інтерфейсом Speech API, розробленим компанією Microsoft.

Синтезатор мови Google дозволяє озвучувати текст в додатках. Наприклад, він використовується:

- в Google Play Книгах для читання вголос;
- в Google Перекладач для вимови слів;
- в TalkBack і інших сервісах спеціальних можливостей для озвучування тексту на екрані;
- в інших додатках, які можна знайти в Play Маркеті.

SpeechText – технологія, яка розроблена спеціально, для тих, у кого зайняті руки, кому цікаво знати вимова будь-яких слів. Цей додаток вимовляє будь-який введений вами текст, на будь-якому з доступних мов. Є можливість зберігати в вигляді звукового файлу на карту пам'яті те, що було написано.

Болтун – зручний додаток для пристроїв на Андроїд. Може озвучити текст SMS, повідомлення електронної пошти, статтю в браузері - будь-які тексти, які ви скопіювали в буфер обміну або введете прямо в додаток.

Висновки до розділу

У цьому розділі було досліджено історію виникнення та розвитку систем розпізнавання мови. По сучасним тенденціям можна зробити

висновок, що задача є дуже актуальною. Розглянуто основні ознаки, за якими можна класифікувати системи розпізнавання людської мови і те, як ця ознака може впливати на роботу системи. Був наведений порівняльний аналіз декількох найбільш відомих та кращих існуючих систем. Окремо було досліджено синтезатори мови, а саме моделі синтезу мови та порівняльний аналіз існуючих платформ.

Глава 2. Структура мовлення та архітектура розпізнавання, загальні поняття про нейронну мережу.

2.1 Структура мовлення

Мовлення - явище складне. Люди рідко розуміють, як вона виробляється та сприймається. Наївне сприйняття часто полягає в тому, що мова побудована зі слів, і кожне слово складається з складів. Реальність, на жаль, дуже відрізняється. Мовлення - це динамічний процес без чітко виділених частин. Завжди корисно отримати звуковий редактор, заглянути в запис промови та прослухати її. Нижче буде представлений процес розпізнавання від подачі звукової хвилі і до кінцевого рішення, тобто запису тексту.

Схематично процес розпізнавання мови можна представити таким чином, як на рисунку 2.1:

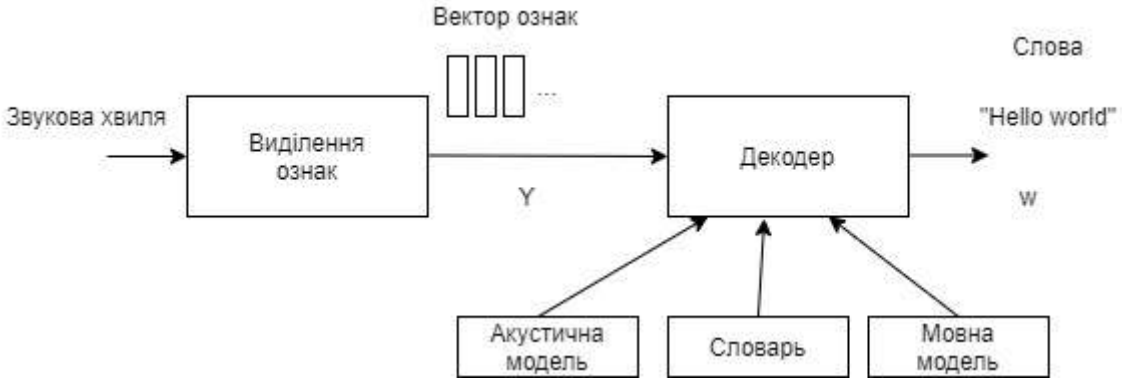


Рисунок 2.1 – Процес розпізнавання мови

2.2 Архітектура систем розпізнавання

Етап вилучення ознак має на меті забезпечити компактне зображення форми мовної хвилі. Тобто у цифровому сигналі виділяються ділянки, які містять розмовну частину, відсіюються проміжні, а також знаходяться параметри ознак мовлення. Основна ціль даного етапу полягає в тому, щоб

зберегти корисні дані та відсіяти непотрібну інформацію, фоновий шум, емоції та інше.

Вектори ознак, як правило, розбиті на фрейми довжиною приблизно 25 мс і обчислюються кожні 10 мс. Одна з найпростіших і найбільш широко застосовуваних схем кодування заснована на мел-частотних кепстральних коефіцієнтах (Mel-Frequency Cepstral Coefficient - MFCC). Основне значення для розуміння мови полягає в тому, що звуки, що утворюються людиною, відрізняється за формою голосового тракту, включаючи язик, зуби тощо. Ця форма визначає, який звук виходить. Якщо ми можемо точно визначити форму, це повинно дати нам точне уявлення про вироблену фонему. Форма голосового тракту проявляється в оболонці короткого часу енергетичного спектру, а завдання MFCC - точно представляти цю оболонку.

Мел-частотні кепстральні коефіцієнти (MFCC) - це функція, яка широко використовується в автоматичному розпізнаванні мови та динаміків. Вона були представлені Девісом і Мермельштейном у 1980-х роках і з тих пір є найсучаснішою. До впровадження MFCC, лінійні коефіцієнти прогнозування (LPC) та лінійні коефіцієнти прогнозування (LPCC) були головними функціями для автоматичного розпізнавання мови. Далі викладені основні аспекти мел-частотних кепстральних коефіцієнтів, а саме як їх реалізувати та чому вони є особливими для розпізнавання.

Алгоритм знаходження мел-частотних кепстральних коефіцієнтів для заданого фрейму наступний:

1. Розкладаємо у ряд Фур’є.

Вихідний мовний сигнал запишемо в дискретному вигляді як:

$$x[n], 0 \leq n \leq N, \quad (1)$$

Error! No sequence specified.

Застосуємо до нього перетворення Фур’є, а саме можна використати швидке перетворення Фур’є (FFT реалізація):

$$X_a[k]\sum_{n=0}^{N-1} x[n]e^{\frac{-2\pi i}{N}kn}, 0 \leq k \leq N, \quad (2)$$

Далі застосуємо віконну функцію Хеммінга для зглажування значень на межах фреймів:

$$H[k] = 0.54 - 0.46\cos\left(\frac{2\pi k}{N-1}\right), \quad (3)$$

В результаті отримуємо наступний вектор:

$$X[k] = X[k] * H[k], 0 \leq k < N, \quad (4)$$

Важливо розуміти, що після цього перетворення по осі X ми маємо частоту (hz) сигналу, а по осі Y - магнітуду (як спосіб піти від комплексних значень). Нижче на рисунку 2.2 наведені частота і магнітуда сигналу:

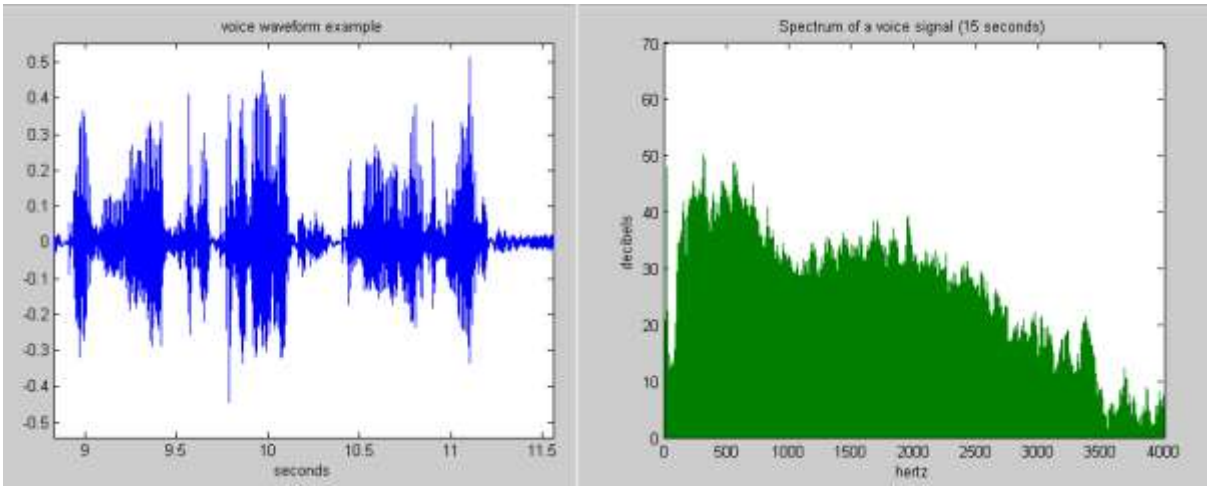


Рисунок 2.2 – Частота та магнітуда сигналу

2. Розкладаємо спектр отриманий вище по mel-шкалі.

Мел – це одиниця висоти звуку, заснована на сприйнятті цього звуку нашими органами слуху. Як відомо, амплітудно-частотна характеристика людського вуха навіть віддалено не нагадує пряму, і амплітуда - не зовсім точна міра гучності звуку. Тому і ввели емпірично підібрані одиниці гучності, наприклад, фон. В першу чергу мел одиниця залежить від частоти звуку (а також від гучності і тембру), як показано на рисунку 2.3.

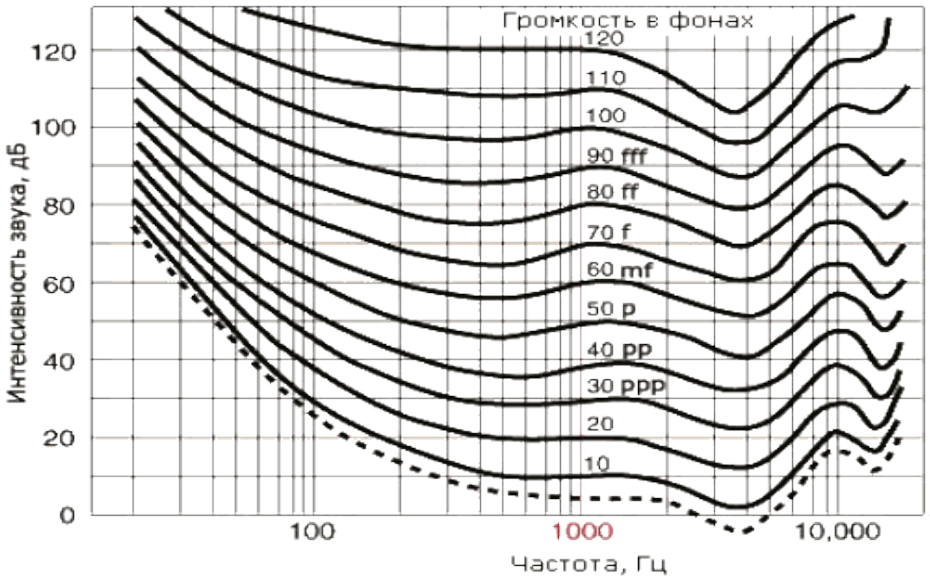


Рисунок 2.3 – АЧХ вуха людини

Аналогічно, сприймається людським слухом висота звуку не зовсім лінійно залежить від його частоти, як показано на рисунку 2.4.

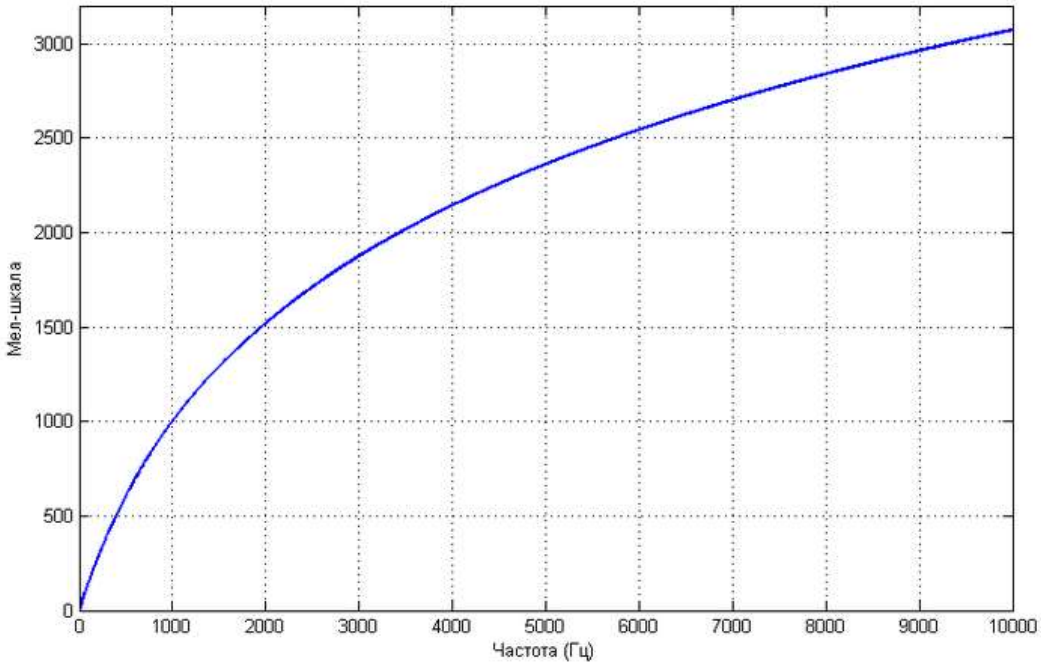


Рисунок 2.4 – Графік залежності mel-одиниць від частот

Така залежність не претендує на велику точність, але зате виконується наступною формулою:

$$M(f) = 1127 * \log\left(1 + \frac{f}{700}\right), \quad (5)$$

Обернене перетворення має наступний вигляд:

$$M^{-1}(m) = 700 * \left(\exp\left(\frac{m}{1125}\right) - 1\right), \quad (6)$$

Подібні одиниці виміру часто використовують при вирішенні задач розпізнавання, так як вони дозволяють наблизитися до механізмів людського сприйняття, яке поки що лідирує серед відомих систем розпізнавання мови.

Для розкладання спектру сигналу по mel-шкалі необхідно побудувати mel-фільтри. Мел-фільтр є трикутним віконною функцією, яка підсумовує енергію на своєму діапазоні частот і обчислює mel-коефіцієнти. Знаючи кількість mel-коефіцієнтів і діапазон частот можна побудувати набір наступних фільтрів, наприклад, як на рисунку 2.5:

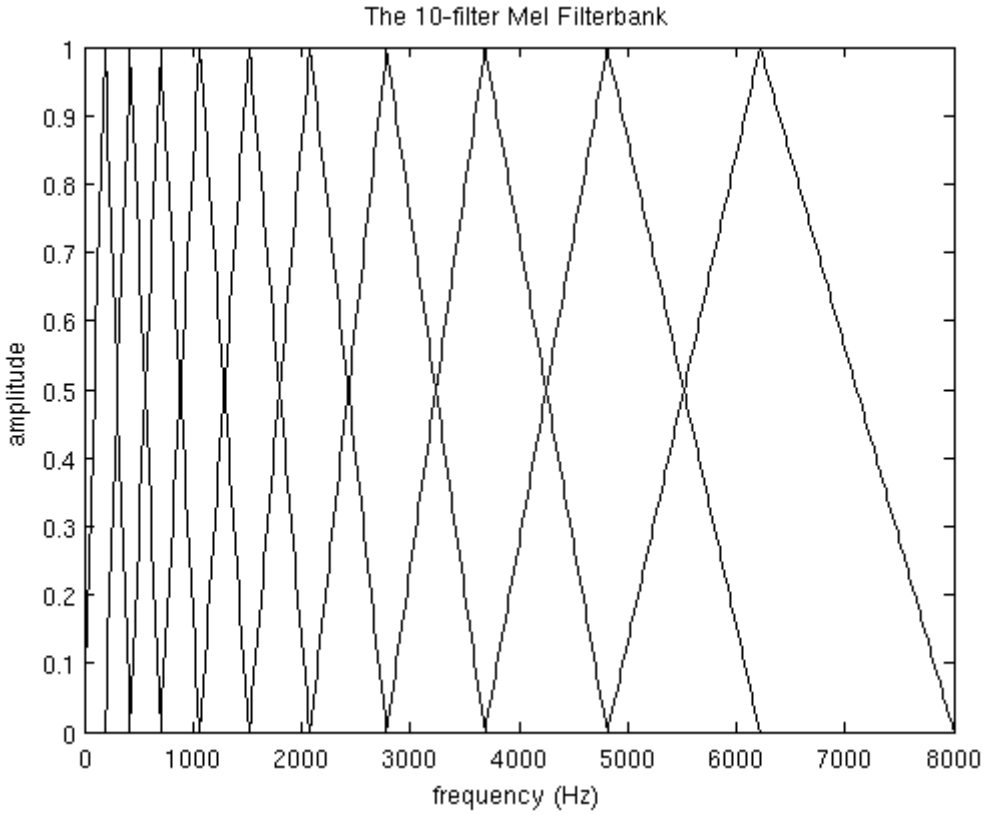


Рисунок 2.5 – Графік mel-фільтрів

Чим більше порядковий номер мел-коефіцієнта, тим ширше основа фільтра. Це пов'язано з тим, що розбиття потрібному нам діапазону частот на оброблювані фільтрами діапазони відбувається на шкалі Мілове.

Нехай ми маємо фрейм з 256 елементами з частотою звуку 16000 Hz. Припустимо, що людська мова лежить в діапазоні від 300 до 8000 Hz. Кількість мел-коефіцієнтів, які ми хочемо знайти, рівна 10.

Застосувавши формулу (5) діапазон [300;8000] Hz зменшився до [401.25; 2834.99] Hz.

Для побудови 10 трикутних фільтрів необхідно взяти 12 опорних точок:

m[i] = [401.25, 622.50, 843.75, 1065.00, 1286.25, 1507.50, 1728.74, 1949.99, 2171.24, 2392.49, 2613.74, 2834.99]

Важливо, що точки на mel-шкалі мають рівномірний розподіл.

Застосуємо формулу (6) для оберненого перетворення шкали в Hz.

h[i] = [300, 517.33, 781.90, 1103.97, 1496.04, 1973.32, 2554.33, 3261.62, 4122.63, 5170.76, 6446.70, 8000]

Бачимо, що шкала поступово розтягується, вирівнюючи тим самим динаміку зростання "значущості" на низьких і високих частотах.

Накладемо отриману шкалу на спектр нашого фрейму. По осі X у нас знаходиться частота. Довжина спектра 256 елементів, при цьому в ньому вміщується 16000 Hz. Обрахувавши пропорцію отримуємо наступну формулу:

$$f(i) = floor\left((frameSize + 1) * \frac{h(i)}{sampleRate}\right), \quad (7)$$

В нашому прикладі отримуємо такий результат:

f(i) = 4, 8, 12, 17, 23, 31, 40, 52, 66, 82, 103, 128

Знаючи опорні точки на осі X спектру, легко побудувати необхідні фільтри за такою формулою:

$$H_m(k) = \begin{cases} 0, & k < f(m-1) \\ \frac{k - f(m-1)}{f(m) - f(m-1)}, & f(m-1) \le k \le f(m) \\ \frac{f(m+1) - k}{f(m+1) - f(m)}, & f(m) \le k \le f(m+1) \\ 0, & k > f(m+1) \end{cases}, \quad (8)$$

3. Фільтруємо та логарифмуємо енергію спектра.

Застосування фільтру полягає в попарном перемножуванні його значень зі значеннями спектра. Результатом цієї операції є мел-коефіцієнт. Оскільки фільтрів у нас M, коефіцієнтів буде стільки ж.

$$S[m] = \log(\sum_{k=0}^{N-1}|X[k]|^2 * H_m[k]), 0 \leq m \leq M, \quad (9)$$

Однак, нам потрібно застосувати мел-фільтри не до значень спектра, а до його енергії. Після чого прологарифмувати отримані результати. Вважається, що таким чином знижується чутливість коефіцієнтів до шумів.

4. Застосовуємо дискретне косинусе перетворення.

За допомогою дискретного косинусного перетворення (DCT) можна отримати "кепстральні" коефіцієнти.

Треба трохи розповісти і про друге слово в назві - кепстра. Мова являє собою акустичну хвилю, яка випромінюється системою органів: легкими, бронхами і трахеєю, а потім перетворюється в голосовому тракті. Якщо припустити, що джерела збудження і форма голосового тракту відносно незалежні, мовний апарат людини можна представити у вигляді сукупності генераторів тональні сигнали і шумів, а також фільтрів:

1. Генератор імпульсної послідовності (тонів)
2. Генератор випадкових чисел (шумів)
3. Коефіцієнти цифрового фільтра (параметри голосового тракту)
4. Нестаціонарний цифровий фільтр

Основна задача дискретного косинусного перетворення – це "стиснення" отриманих результатів, причому з підвищенням значущості перших коефіцієнтів і зменшення значущості останніх. DCT рахуємо за такою формулою:

$$C[l] = \sum_{m=0}^{M-1} S[m] * \cos(\pi * l * \frac{m+\frac{1}{2}}{M})),0 \leq l \leq M, \quad (10)$$

У результаті для кожного фрейму ми отримаємо набір MFCC коефіцієнтів розміру M. Надалі їх можна буде використати у якості вхідних даних для акустичної моделі.

Відповідно до структури мовлення, для розпізнавання мови використовується три моделі:

1. Акустична модель - це функція, що приймає на вхід невелику ділянку акустичного сигналу (кадр або фрейм) і видає розподіл ймовірностей різних фонем на цьому кадрі. Таким чином, акустична модель дає нам можливість по звуку відновити, що було вимовлено - з тим або іншим ступенем впевненості. Фонема - елементарна одиниця людської мови.
2. Мовна модель використовується для обмеження пошуку слів. Вона визначає, яке слово могло б слідувати раніше розпізнаним словам (відповідність є послідовним процесом) і допомагає значно обмежити процес узгодження шляхом вилучення слів, які не вірогідні. Найпоширеніші мовні моделі - це n-грамові мовні моделі - вони містять статистику послідовностей слів – і моделі кінцевих мов - вони визначають мовленнєві послідовності методом автоматичного обмеження стану, іноді з вагами. Щоб досягти хорошого показника точності, мовна модель повинна бути дуже успішною в обмеженні простору пошуку. Це означає, що слід дуже добре передбачити наступне слово. Мовна модель, як правило, обмежує словниковий запас, який вважається словами, який він містить. Це проблема розпізнавання імен. Щоб вирішити це, мовна модель може містити менші фрагменти, як підслова або навіть фонеми. Обмеження шуканого фрагмента в цьому випадку зазвичай гірше, а відповідні точності розпізнавання нижчі, ніж у мовній моделі на основі слів.

3. В ході роботи системи автоматичного розпізнавання мови завдання розпізнавання зводиться до визначення найбільш вірогідною послідовності слів, відповідних змісту мовного сигналу. Найбільш ймовірний кандидат повинен визначатися з урахуванням як акустичної, так і лінгвістичної інформації. Це означає, що необхідно проводити ефективний пошук серед можливих кандидатів з урахуванням різної ймовірнісної інформації. При розпізнаванні суцільної мови число таких кандидатів величезне, і навіть використання найпростіших моделей призводить до серйозних проблем, пов'язаних з швидкодією і пам'яттю систем. Як результат, це завдання виноситься в окремий модуль системи автоматичного розпізнавання мови, званий декодером. Декодер повинен визначати найбільш граматично ймовірну гіпотезу для невідомого висловлювання - тобто визначати найбільш ймовірний шлях по мережі розпізнавання, що складається з моделей слів (які, в свою чергу, формуються з моделей окремих фонів). Правдоподібність (likelihood) гіпотези визначається двома факторами, а саме можливостями послідовності фонів, що приписуються акустичної моделлю, і можливостями проходження слів друг за другом, обумовленими моделлю мови.

   Ці три об'єднання об'єднані разом в двигун для розпізнавання мови.

2.3 Методи та алгоритми розпізнавання мови

Сучасні методи та алгоритми розпізнавання мовлення можна поділити на наступні класи:

- Динамічне програмування - алгоритм динамічної трансформації часової шкали (Dynamic Time Wrapping - DTW).
- Приховані Марківські моделі (Hidden Markov Models - HMM).

– Нейронні мережі (Neural Networks - NN).

У багатьох статтях, які відносяться до розпізнавання мови довгий час базовим підходом вважалися приховані марківскі моделі, проте останнім часом активно розвиваються методи, засновані на застосуванні нейронних мереж. Далі наводиться більш детальний огляд зазначених методів.

2.3.1 Динамічне програмування

Визначення слова може здійснюватися шляхом порівняння числових форм сигналів або шляхом порівняння спектрограми сигналів. Процес порівняння в обох випадках повинен компенсувати різні довжини послідовності і нелінійний характер звуку. DWT алгоритму вдається розібрати ці проблеми шляхом знаходження деформації, відповідної оптимальному відстані між двома рядами різної довжини.

Існують 2 особливості застосування алгоритму:

1. Пряме порівняння числових форм сигналів. В цьому випадку, для кожної числової послідовності створюється нова послідовність, розміри якої значно менше. Алгоритм має справу з цими послідовностями. Числова послідовність може мати кілька тисяч числових значень, в той час як підпослідовність може мати кілька сотень значень. Зменшення кількості числових значень може бути виконано шляхом їх видалення між кутовими точками. Цей процес скорочення довжини числової послідовності не повинен змінювати свого уявлення. Безсумнівно, процес призводить до зменшення точності розпізнавання. Однак, беручи до уваги збільшення швидкості, точність, по суті, підвищується за рахунок збільшення слів в словнику.

2. Подання сигналів спектрограм і застосування алгоритму DTW для порівняння двох спектрограм. Метод полягає в поділі цифрового сигналу на деяку кількість інтервалів, які будуть перекриватися. Для кожного

імпульсу, інтервали дійсних чисел (звукових частот), буде розраховувати Швидким перетворення Фур'є, і буде зберігатися в матриці звуковий спектрограми. Параметри будуть однаковими для всіх обчислювальних операцій: довжин імпульсу, довжини перетворення Фур'є, довжини перекриття для двох послідовних імпульсів. Перетворення Фур'є є симетрично пов'язаних з центром, а комплексні число з одного боку пов'язані з числами з іншого боку. У зв'язку з цим, тільки значення з першої частини симетрії можна зберегти, таким чином, спектрограмма представлятиме матрицю комплексних чисел, кількість ліній в такій матриці є рівною половині довжини перетворення Фур'є, а кількість стовпців буде визначатися в залежності від довжини звуку. DTW буде застосовуватися на матриці дійсних чисел в результаті сполучення спектрограми значень, така матриця називається матрицею енергії.

DTW алгоритми є дуже корисними для розпізнавання окремих слів в обмеженому словнику. Для розпізнавання швидкого мовлення використовуються приховані моделі Маркова. Використання динамічного програмування забезпечує поліміальную складність алгоритму: O (n2v), де n - довжина послідовності, а v кількість слів у словнику.

DWT мають кілька слабких сторін. По-перше, O (n2v) складність не задовольняє великим словників, які збільшують успішність процесу розпізнавання. По-друге, важко вирахувати два елементи в двох різних послідовностях, якщо взяти до уваги, що існує безліч каналів з різними характеристиками. Проте, DTW залишається простим в реалізації алгоритмом, відкритим для поліпшень і відповідним для додатків, яким потрібна проста розпізнавання слів: телефони, автомобільні комп'ютери, системи безпеки і т.д.

2.3.2 Прихована марківська модель

Прихована марківська модель (ПММ) представляє з себе систему, яка імітує випадковий процес, еволюція якого залежить лише від поточного стану моделі і не залежить від попередньої історії. У такому випадку виникає задача з наближенням значень прихованих (невідомих) моделей із заданою послідовністю спостереження значень.

Таким чином, марківський процес може зобразити як на рисунку 2.6:

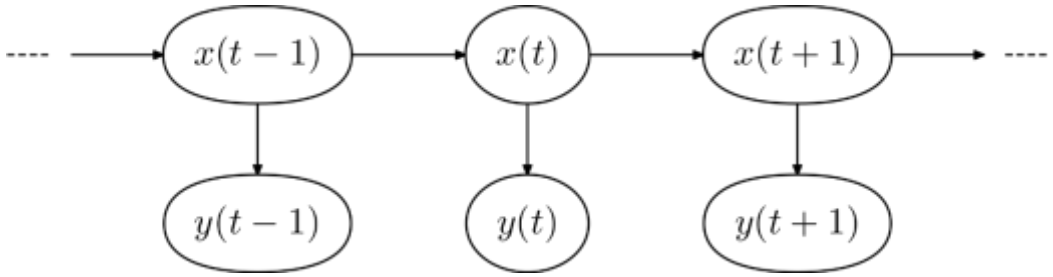


Рисунок 2.6 – Марківський процес

Тут $x_t \in \{1, \dots, N\}$ - прихований стан моделі в момент часу t, $y_t \in R^d$- змінна, яка спостерігається в момент часу t. Значення y_t залежить тільки від x_t, а x_t тільки від стану в момент часу t - 1, тобто від x_{t-1}. Для подальшого задання моделі потрібно визначити матрицю ймовірностей переходів:

$$a \in R^{N \times N} | a_{i,j} = p(x_t = j | x_{t-1} = i),$$

розподілу спостережуваних змінних для кожного стану $p(y_t|x_t)$ і розподіл ймовірностей початкових станів $p(x_1)$. Зазвичай розподіл:

$$p(y_t|x_t) = b_t \, ,$$

роблять нормальним:

$$b_t(y) = \mathcal{N}(y; \mu_t, \sum_t),$$

Форма вхідної звукової хвилі від мікрофона перетворюється в послідовність акустичних векторів фіксованого розміру $Y = \{y_1,\ y_2, \dots\ y_T\}$, в процесі, який називається виділенням ознак. Декодер потім намагається знайти послідовність слів $w = \{w_1,\ w_2, \dots\ w_L\}$, якими з найбільшою ймовірністю згенерована Y, тобто представимо наступним чином:

$$\widehat{w} = \arg\max_{w}\{P(w|Y)\},$$

Часто незручно працювати з ймовірністю $P(w|Y)$, тому, використовуючи формулу Байеса, її замінюють на наступним чином:

$$argmax_w\left\{\frac{P(w|Y)*P(w)}{P(Y)}\right\} = argmax_w\{P(w|Y) * P(w)\},$$

Тут $P(Y|w)$ визначається акустичної моделлю, а P (w) – мовною. Кожному слову відповідає впорядкована послідовність фонем (можливо не одна, оскільки у слові може бути кілька правильних вимов). Акустична модель для конкретного слова виходить конкатенацією моделей для окремих фонем, які, в свою чергу, навчаються на розміченій тренувальної вибірці, наприклад, за допомогою алгоритму прямого-зворотного ходу. Основною одиницею мовного потоку, яка представляється акустичною моделлю є звук. Наприклад, слово “cat” складається з трьох звуків /c/ /ae/ /t/. Близько 40 таких звуків використовуються в англійській мові. Для будь-якого даного w відповідна акустична модель синтезується за допомогою об'єднання звуків, щоб скласти слова, які визначені словником. Далі

визначається найбільш ймовірну послідовність слів, яка і є кінцевим результатом розпізнавання.

2.3.3 Нейронні мережі

Особливе місце в задачі розпізнавання мови займають методи, засновані на нейромережевих технологіях. У цих методах результат розпізнавання є продуктом функціонування нейронної мережі певного виду і топології. Нейронні мережі являють собою безліч пов'язаних між собою елементарних процесорів (нейроподібних елементів), кожний з яких виконує відносно прості функції.

Прототипом нейрона є біологічна нервова клітина. Нейрон складається з тіла клітини, або соми, і двох типів зовнішніх деревоподібних гілок: аксона і дендритів. Тіло клітини включає ядро, яке містить інформацію про спадкові властивості, і плазму, що володіє молекулярними засобами для виробництва і передачі елементів нейрона необхідних йому матеріалів. Нейрон отримує сигнали (імпульси) від інших нейронів через дендрити і передає сигнали, згенеровані тілом клітки, вздовж аксона, що наприкінці розгалужується на волокна, на закінченнях яких знаходяться синапси. Математична модель нейрона описується наступним співвідношенням:

$$y = f(s), s = \sum_{i=1}^{n} x_i w_i + b,$$

де w_i – вага синапса;

b – значення зміщення;

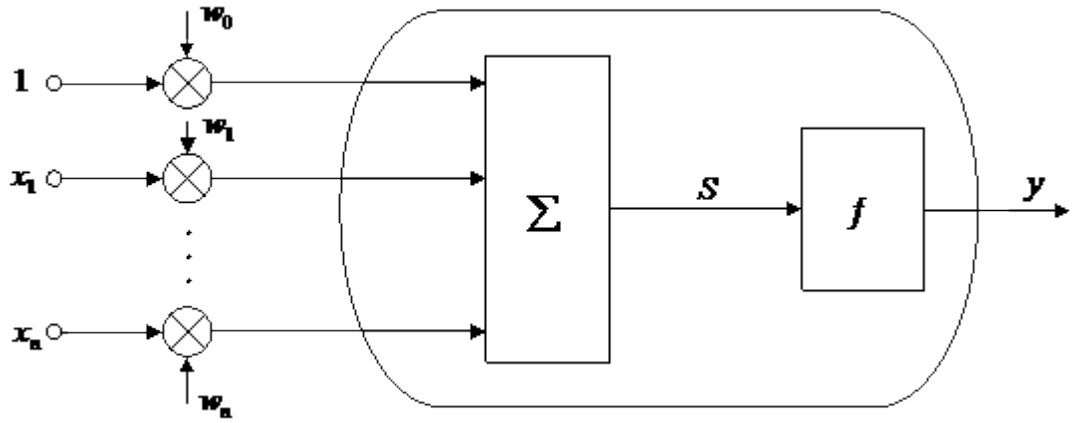
s – вхідний сигнал;

y – вихідний сигнал нейрона;

n – число входів нейрона;

f - функція активації.

Структурна схема нейрона подана на рисунку 2.7:



де x_1, x_2, ..., x_n - вхідний сигнал нейрона;

w_1, w_2, ..., w_n - набір вагових коефіцієнтів;

f (s) - функція активації;

y - вихідний сигнал.

Рисунок 2.7 – Структурна схема нейрона

Як видно, нейроподібні елемент виконує нескладні операції зваженого підсумовування, обробляючи результат нелінійним пороговим перетворенням. Особливість нейросетевого підходу полягає в тому, що структура з простих однорідних елементів дозволяє вирішувати нетривіальні завдання завдяки складній організації зв'язків між елементами. Структура зв'язків визначає функціональні властивості мережі в цілому.

Процес функціонування мережі залежить від величин синаптичних зв'язків. Задавши певну структуру мережі (на етапі проектування), знаходять оптимальні значення вагових коефіцієнтів w_1, w_2, ..., w_n і зсувів b всіх нейронів. Цей етап називають навчанням нейронної мережі.

Вирішити поставлене завдання розпізнавання мови за допомогою нейронної мережі заданої структури - означає шляхом навчання за вибіркою, заданої k парами значень вхідних і вихідних векторів (X^i, Y^i), i=1,…,k, знайти таку конфігурацію, щоб забезпечити найбільш оптимальне в певному сенсі її функціонування. Розрізняють два підходи до навчання нейронної мережі: навчання з учителем і навчання без вчителя (самонавчання). При навчанні з учителем на вхід мережі подається один з векторів X^i навчальної вибірки, вихід мережі Y порівнюється з виходом Y^i навчальної вибірки, і при необхідності робляться поправки в вагові коефіцієнти і зміщення нейронів мережі. В алгоритмах навчання без учителя підстроювання ваг синапсів проводиться на підставі інформації про стан нейронів і вже наявних вагових коефіцієнтів по одному з правил навчання. Процес повторюється, поки вихідні значення мережі не стабілізуються із заданою надійністю.

Нейрони можуть групуватися в мережеву структуру по-різному. Функціональні особливості нейронів і спосіб їх об'єднання в мережеву структуру визначають особливості нейромережі. Для вирішення задач ідентифікації та управління найбільш адекватними є багатошарові нейронні мережі (БНС) прямої дії, або багатошарові персептрони. При проектуванні БНС нейрони об'єднують в шари, кожен з яких обробляє вектор сигналів від попереднього шару. Мінімальною реалізацією є двошаровий нейронна мережа, що складається з вхідного (розподільного), проміжного (прихованого) і вихідного шару, як на рисунку 2.8.

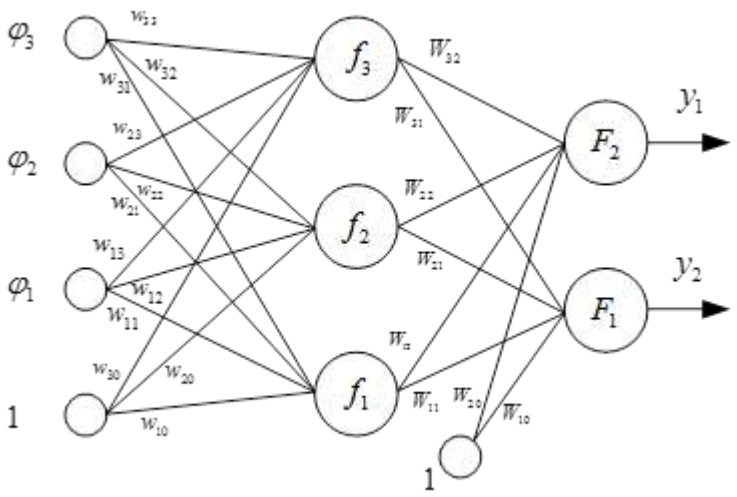


Рисунок 2.8 – Структурна схема двошарової нейронної мережі

Реалізація моделі двошарової нейронної мережі прямого дії має наступний математичний вигляд:

$$y(\theta) = F_i\left(\sum_{j=1}^{n_h} W_{ij} f_j\left(\sum_{j=1}^{n_h} w_{ij}\varphi_j + w_{j0}\right) + W_{j0}\right)$$

n_φ – розмірність вектора входів φ нейронної мережі;

θ – вектор параметрів, що настроюються нейронної мережі, що включає вагові коефіцієнти і нейронні зміщення (w_{ij}, W_{ij});

$f_j(x)$ – активаційна функція нейронів прихованого шару;

$F_i(x)$ – активаційна функція нейронів вихідного шару.

Найважливішою відмінною рисою нейромережевого методу є можливість паралельної обробки. Дана особливість при великій кількості міжнейронних зв'язків дає можливість досягти значного прискорення процесу обробки даних. У багатьох випадках з'являється можливість обробки мовних сигналів в реальному часі. Ще один важливий плюс в нейромережевому методі - це узагальнення отриманих знань. Нейронна мережа має ті якості, які властиві для так званого штучного інтелекту.

В результаті процесу навчання нейронна мережа здатна виявляти складні залежності між вхідними та вихідними даними. При узагальненні інформації мережа дозволить повернути вірний результат на підставі неповних або перекручених даних. При великій кількості з'єднань між нейронами мережа набуває стійкість до помилок, які виникають на деяких лініях. Роботу пошкоджених зв'язків беруть на себе справні лінії, в результаті чого робота мережі не зазнає істотних змін.

Процедура навчання мережі є досить трудомістким в обчислювальному плані процесом, вимагає великого розміру навчальної вибірки. Крім того, процедура навчання не у всіх випадках гарантує отримання результату, багато робіт присвячено проблемам навчання нейронних мереж. Незважаючи на ці недоліки даного методу, він є одним з найчастіше використовуваних для розпізнавання мови, володіє наступними перевагами перед іншими методами: є нелінійним, досить стійкий до зашумлення мовного сигналу, легко піддається розпаралеливанню обчислень.

Розглянемо «класичний варіант» багатошарової мережі, де синаптичні зв'язки можуть визначатися будь-якими дійсними числами, а вихід нейрона - дійсними числами з інтервалу від 0 до 1. Як активаціонну функцію використовуємо сигмоїд, число шарів – довільне.

1. Визначаємо M матриць вагових коефіцієнтів W розміром NxN, де M – число шарів, N – число нейронів в одному шарі. $W_{i,j,k}$ буде позначати вагу j-го входу k-го нейрона в i-м шарі. Ініціалізуємо матриці деякими малими випадковими (не однаковими) значеннями.
2. Подаємо на входи мережі певні значення X, для яких відомі правильні значення виходів мережі Y*.
3. Обчислюємо значення виходів мережі для поточного стану матриць W. Тобто для вхідного вектора X обчислюється вихідний вектор Y. Для цього необхідно послідовно обчислити вихід для кожного шару

мережі з першого по останній. Для і-го шару в векторному вигляді це можна записати так:

$$O_i = F(XW_i), \text{ якщо і – перший шар;}$$
$$O_i = F(O_{i-1}W_i), \text{ якщо і – не перший шар,}$$

O_i – вектор виходу і-го шару;

F – активаційна функція;

X – вектор входів;

O_{i-1} – вектор виходу (і-1)-го шару;

W_i – матриця вагових коефіцієнтів і-го шару.

4. Обчислюємо вектор

$$\Delta Y = Y\text{-}Y^*$$

5. Якщо ΔY менше заданої похибки, переходимо до кроку 9.
6. Для шару з номером M (тобто в останньому шарі) виробляємо наступні операції:
   6.1. Для всіх нейронів в шарі з номера 1 по N робимо наступні операції:
      6.1.1. Для всіх ваг нейрона з номера 1 по N робимо наступні операції:
         6.1.1.1. Розраховуємо вектор

$$\delta M = X(1 - X)\Delta Y$$

         6.1.1.2. Розраховуємо величину

$$\Delta W_{i,j,k} = \eta\delta i, kOi - 1, j,$$

де η – коефіцієнт швидкості навчання (від 0.01 до 1.0).

6.1.1.3. Коригуємо величину вагового коефіцієнта, додаючи до $W_{M,j,k}$ величину $\Delta W_{M,j,k}$.

7. Для шарів з номером M-1 по перший послідовно проводимо такі операції:

7.1. Для всіх нейронів в шарі з номера 1 по N робимо наступні операції:

7.1.1. Для всіх ваг нейрона з номера 1 по N робимо наступні операції:

7.1.1.1. Розраховуємо вектор

$$\delta_i = O_{i+1}(1 - O_{i+1})[\sum \delta_{i+1,j} W_{i+1,j,k}]$$

7.1.1.2. Розраховуємо величину

$$\Delta W_{i,j,k} = \eta \delta i, kOi - 1, j$$

де η– коефіцієнт швидкості навчання (від 0.01 до 1.0).

7.1.1.3. Коригуємо величину вагового коефіцієнта, додаючи до $W_{M,j,k}$ величину $\Delta W_{M,j,k}$.

8. Перехід до кроку 3.

9. Кінець (навчання закінчено).

Описаний алгоритм застосовується достатню кількість разів, щоб всі варіанти вихідних значень могли правильно виходити при завданні довільних значень входу із заданою вірогідністю помилки.

З усього вищесказаного можна зробити висновок про ефективність застосування нейромережевого методу до розглянутої задачі розпізнавання

мови. Використання нейромережевих методів для розпізнавання мови дозволяє істотно підвищити точність системи розпізнавання мовного сигналу і збільшити її швидкодію.

Висновки до розділу

В цьому розділі ми детально розглянули крок за кроком архітектуру розпізнавання мови. В першому етапі виділення ознак було детально розписано алгоритм знаходження мел-частотних кепстральних коефіцієнтів (MFCC) для заданого фрейму.

Наступний етап декодера включає в себе наведені три моделі: акустична, мовна та словник, які об'єднані разом. Детально оглянуто сучасні методи та алгоритми розпізнавання мовлення (Dynamic Time Wrapping, Приховані Марківські моделі, Нейронні мережі). Особливу увагу було приділено методу, заснованого на нейронні мережах, а саме детально розглянуто процедуру навчання «класичного варіанту» багатошарової мережі.

Глава 3. Опис програми реалізації розпізнавання мови.

2.4 Мова програмування та середовище розробки

В якості мови програмування було обрано мову Java, у цієї мови є багато переваг перед іншими мовами програмування, що дозволяє вирішувати з її допомогою практично будь-які завдання.

Характерні особливості мови програмування Java:

- багатопоточна обробка;
- простота;
- об’єктно-орієнтованість;
- розподіленість;
- безпека;
- інтерпритованість;
- незалежність від архітектури;
- високопродуктивність;
- динамічність.

Менеджер захисту персональних файлів розроблюється в середовищі програмування NetBeans, яке є вільним інтегрованим середовищем розробки для Java. Для розробки було обрано це середовище через його вільність та зручність у використанні.

NetBeans IDE — вільне інтегроване середовище розробки (IDE) для мов програмування Java, JavaFX, C/C++, PHP, JavaScript, HTML5, Python, Groovy. Середовище може бути встановлене і для підтримки окремих мов, і у повній конфігурації. Середовище розробки NetBeans за замовчуванням підтримує розробку для платформ J2SE і J2EE.

Поширюється у сирцевих текстах під ліцензіями GPLv2 і CDDL. Проект NetBeans IDE підтримувався і спонсорувався фірмою Sun Microsystems і після придбання Sun — Oracle, проте розробка NetBeans

ведеться незалежно співтовариством розробників (NetBeans Community) і компанією NetBeans.Org.

NetBeans IDE доступна для платформ Microsoft Windows, GNU/Linux, FreeBSD, і Solaris (як SPARC, так x86). Для інших платформ доступна можливість зібрати NetBeans самостійно із сирцевих текстів.

За якістю і можливостям останні версії NetBeans IDE змагається з найкращим інтегрованими середовищами розробки для мови Java, підтримуючи рефакторинг, профілювання, виділення синтаксичних конструкцій кольором, автодоповнення мовних конструкцій на льоту, шаблони коду та інше.

2.5 Аналіз архітектури системи розпізнавання голосу

Нижче ми розглянемо та проаналізуємо архітектуру системи розпізнавання голосу «Jarvis».

До основних класів задач голосового інтерфейсу системи розпізнавання голосу слід віднести:

- Синтез мовлення – він включає в себе комплекс підзадач, які полягають в перетворенні друкарського тексту у мовний сигнал;
- Аналіз та розпізнавання мовлення – комплекс задач, які включають в себе запис, оцифрування і аналіз мовлення для розпізнавання отриманого мовного сигналу системою;
- Розуміння (інтерпретація) мовлення – це комплекс задач, зв’язаних з аналізом змісту мовленнєвих повідомлень і формування реакції (відповіді) системи, виконання команд.

В нашому випадку розпізнаватись буде визначений набір команд, для цієї задачі найкраще підходить розпізнавання на основі обмеженого словника та граматики запитів. Це дозволить підвищити точність розпізнавання саме для конкретного набору слів та здійснювати це прямо

на пристрої не використовуючи підключення до мережі. Можна використовувати готовий словник для англійської мови, який скачується разом з акустичною моделлю. Однак чим менше словник - тим швидше пошук по ньому і зберігати в оперативній пам’яті словник на 500000 слів не має сенсу, коли використовуватись буде лише декілька.

Для синтезу мовлення було використано синтезатор text-to-speech (TTS) - вільна система синтезу мови, повністю написана на Java. Ця технологія підтримує три голоси, в нашій програмі обрано голос Кевіна.
Система перетворення тексту в мову складається з наступних частин:

- Аналіз тексту (перетворення необробленого тексту, що містить символи як цифри і абревіатури в еквівалент письмовий)
- Підбір звукових елементів (відповідность акустичного зображення юніта для конкретної фонеми та доречність з'єднання двох сусідніх юнітів)
- Акустична обробка (складення в одне ціле підібрані звукові елементи та озвучення результату)

  Нижче схема наведена на рисунку 3.1:

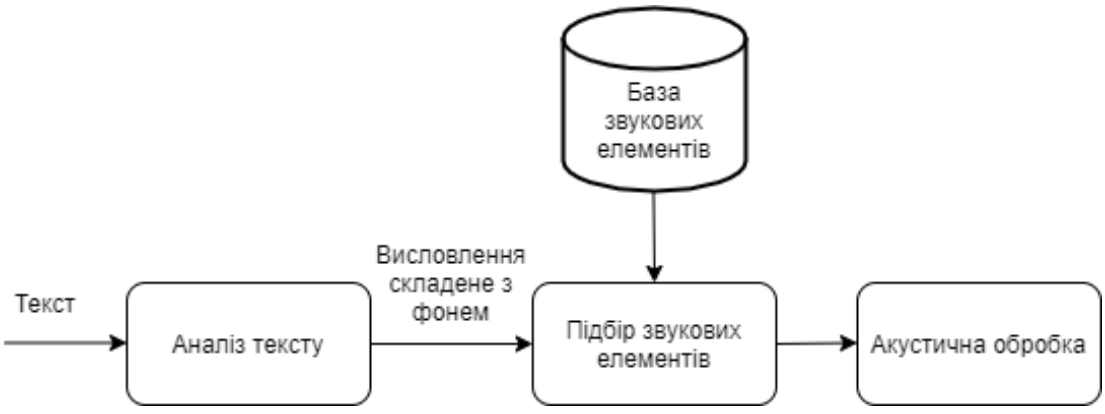


Рисунок 3.1 – Схема синтезу мови

Для досягнення поставлених цілей, було вирішено використати бібліотеку з відкритим кодом Sphynx. Розпізнавання мовлення відбувається

перетворенням мовленнєвого сигналу в текстовий потік. Вхідні дані отримуються з потоку який зчитується мікрофоном та трансформується в масив байтів який можна опрацювати програмою.

Програмний продукт на даному етапі можна представити у вигляді двох частин (packages):

1. Jarvis.cfg – містить конфігураційні файли программи:
   1.1. Commands.dict – словник команд, згенерований на основі наперед визначеного списку команд - структурного (corpus) файлу.
   1.2. En-us.lm – містить n-грамна модель мовну модель (3-грамна - використана для опису команд які пристутні в програмі). Мовна модель описує ймовірності слів і їх комбінацій. Таким чином додаючи певний контекст завдяки якому програма може розпізнавати фрази схожі за своїм звучанням.
2. Jarvis.controller – містить основну бізнес логіку програми а також допоміжні модулі:
   2.1. JarvisSpeechEngineController - основний клас. Ця частина надає конструктор, що приймає в якості аргументу конфігурацію з попередньої частини програми. Система захоплює сигнал мікрофона та перетворює його в масив байтів, який оброблюється базуючись на файлах з попереднього пакету. Якщо перетворений мовленнєвий сигнал відповідає одній з команд, які наперед визначені в програмі, то спрацьовує код, що відповідає за виконання команди, та створюється тестовий масив відповідей якими програма реагує на команди. Використовуючи синтез мовлення обрана навмання фраза з цього масиву дає користувачу зрозуміти що команду було успішно виконано.
   2.2. JavaReplyController – відповідає за генерування відповіді, з можливого масиву відповідей вибирає випадкову і передає на обробку до JarvisTTSController.

2.3. JarvisTTSController – викликає код який синтезує відповідь базуючись на вхідній фразі та перетворює текст на мову.

Нижче наведено словник команд на рисунку 3.2, згенерований на основі наперед визначеного списку команд Commands.dict:

```
ABOUT   AH B AW T
AN      AE N
AN(2)   AH N
ANY     EH N IY
CALL    K AO L
CHECK   CH EH K
CHROME  K R OW M
CLOSE   K L OW S
CLOSE(2)        K L OW Z
CONNECT K AH N EH K T
CORTANA K AA R T AE N AH
COULD   K UH D
DO      D UW
EMPTY   EH M P T IY
EMPTY(2)        EH M T IY
EXCEL   IH K S EH L
FILM    F IH L M
GOODBYE,        G UW D AY
HELP    HH EH L P
I       AY
JARVIS  JH AA R V AH S
JARVIS(2)       JH AA R V IH S
MAIL    M EY L
ME      M IY
ME,     M IY
NEED    N IY D
NOTEPAD N OW T P AE D
OFFICE  AO F AH S
OPEN    OW P AH N
OPERA   AA P R AH
RDP     AA R D IY P
RUN     R AH N
SIRI    S IH R IY
SKYPE   S K AY P
TELL    T EH L
TRASH   T R AE SH
UMBRELLA?       AH M B R EH L AH
WORD    W ER D
YOU     Y UW
```

Рисунок 3.2 – Словник команд

2.6 Інструкція по застосуванню програмного продукту “Jarvis”

Програму можна запустити двома способами:

1. Запускати “Jarvis.jar” (Executable Jar File) через командну строку.

Відкрити командну строку та перейти в директорію, де зберігається файл “Jarvis.jar”, набрати команду java -jar Jarvis.jar, як показано на рисунку 3.3:

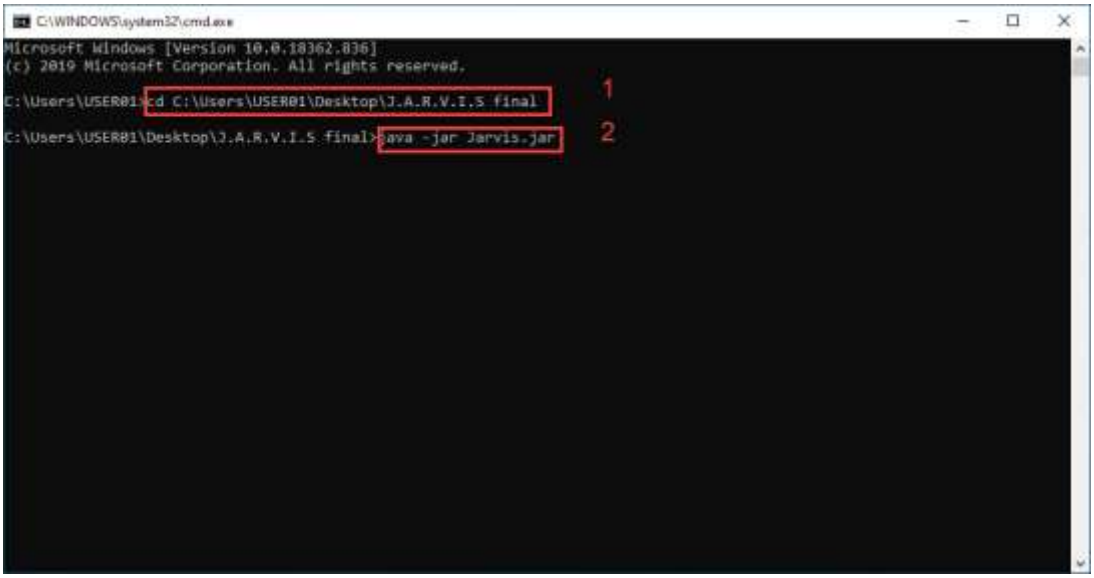


Рисунок 3.3 – Запуск програми через термінал

2. Запускати “Jarvis_launcher.bat” (Windows Batch File).

Перед тим як запускати програму за допомогою “Jarvis_launcher.bat” потрібно вказати шлях до виконуючого файлу “Jarvis.jar” (виконується тільки один раз після скачування програмного продукту). Для цього необхідно відкрити запускаючий файл в редагованому режимі та вказати в лапках шлях, як представлено на рисунку 3.4:

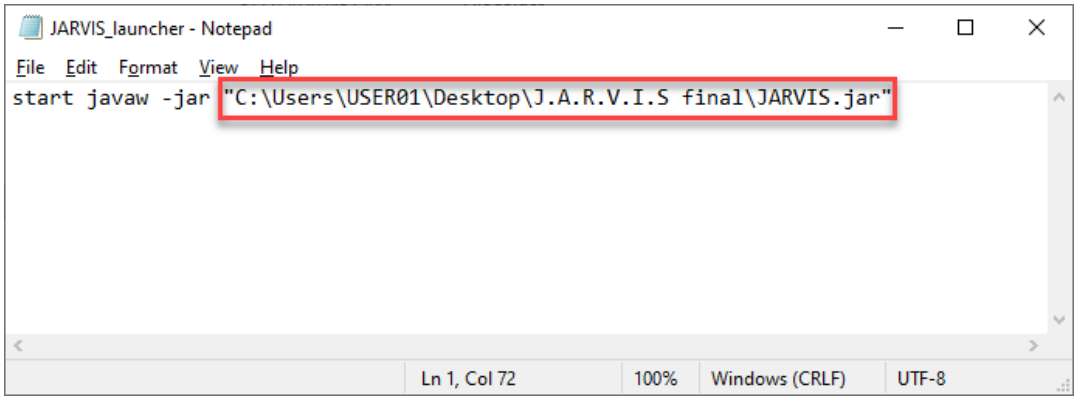


Рисунок 3.4 – Редагований режим файлу “Jarvis_launcher.bat”

Зберегти зміни та запустити “Jarvis_launcher.bat”: програма готова до роботи.

Варто зауважити, що “Jarvis_launcher.bat” може бути з легкістю додано як файл автозавантаження, що надасть змогу запускати програму одночасно з операційною системою.

Висновки до розділу

В цьому розділі було наведено опис програми реалізації розпізнавання мови. Для реалізації програмного продукту було обрано мову програмування Java та середовище розробки NetBeans. Був наведений опис переваг обраних застосунків.

Розглянуто та проаналізовано архітектуру системи розпізнавання голосу «Jarvis». До основних класів задач голосового інтерфейсу системи розпізнавання голосу було віднесено аналіз тексту, підбір звукових елементів та акустична обробка. Програмний продукт представлено у вигляді двох частин (packages) та описано складові файли та роль кожного з них.

Була наведена інструкція по користуванню програмного продукту “Jarvis”.

Глава 4. Опис експериментальних досліджень

В цій главі наведено опис вхідних данних та очікувані результати. Також проведено дослідження якості розробленої системи розпізнавання мови.

4.1 Вхідні та вихідні набори

Нижче наведений перелік вхідних фраз та очікуваний результат у вигляді виконання команди та відповіді.

1. Спершу завжди іде привітання: “Could you help me, Jarvis?”. Далі генерується відповідь з можливого масиву:
    a. "Greetings, master",
    b. "It's nice to hear your voice",
    c. "You remember my name, so I presume you need something"
2. Далі безпосередньо іде сама команда. Програмний продукт може виконувати наступні команди:
    a. "Connect Office RDP" – під ’єднатись до віддаленого робочого комп’ютера.
    Відповідь: "I am now connecting to your work PC"
    b. "Run Opera", "Close Opera", "Run Chrome" та "Close Chrome" – відкрити та закрити браузер Opera та Chrome відповідно.
    Відповіді:
    "Don't forget to clean your browsing history afterwards. We all know what's in there"
    "The Opera is dead. Long live the Chrome"
    "Just a warning. This app consumes so much resources and it is not even AI"
    "Good call. Always liked opera more"

c. "Open Notepad" та "Close Notepad" – запустити на закрити блокнот.
   Відповіді:
   "Feel free to write down all your thoughts"
   "I hope you saved everything. Not that I care."
d. "Call Skype" – відкрити Skype та вибрати кому дзвонити.
   Відповідь: "Just select who to call."
e. "Do I need an umbrella?" – відкрити прогноз погоди.
   Відповідь: "Could have as well just look into the window."
f. "Open any film" – відкрити Netflix.com.
   Відповідь: "Want to watch something. Pick it yourself."
g. "Open Word" та "Close Word" – відкрити та закрити Word.
   Відповідь: "That I can do. Just make sure that you've saved your diploma."
h. "Check Mail" та "Close Mail" – перевірити пошту в додатку Outlook.
   Відповідь: "Workday is over. No more mail."

4.1 Тестування програми

Щоб перевірити якість програми, продукт був протестований наступним чином:

– Програма була відкрита через термінал, де можливо після подання вхідних даних побачити, що розпізнала програма, а також перевірити деяку статистику, а саме швидкість та тривалість аудіо, використану пам’ять. Це показано на рисунку 4.1:

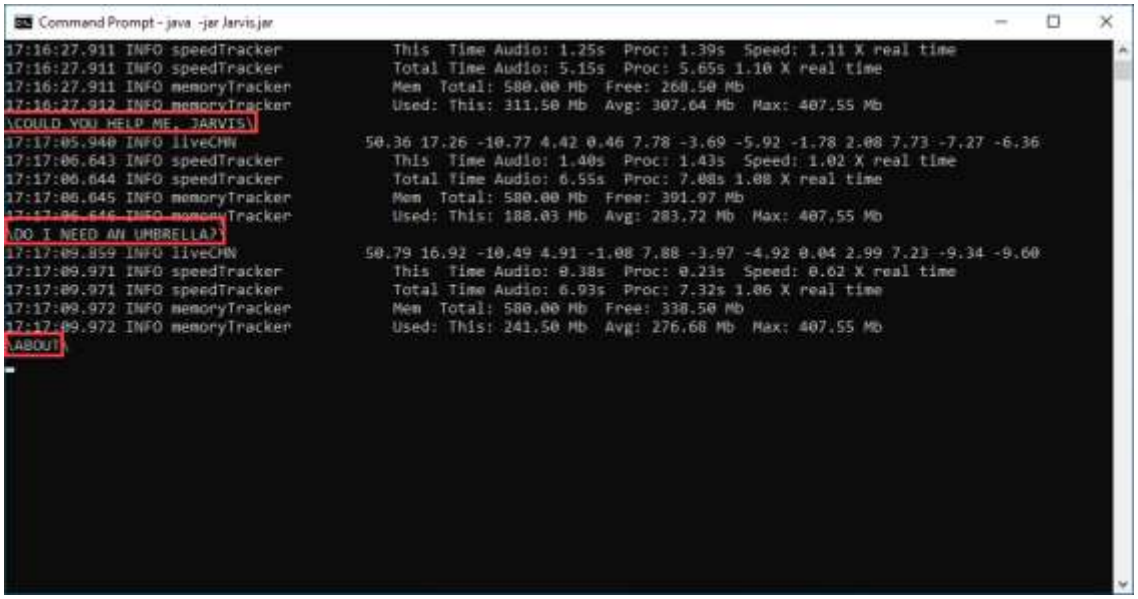


Рисунок 4.1 – Робота програми через термінал

- Програма тестувалась користувачами різної статі та віку (двоє жінок 20 та 30 років; двоє чоловіків 24 та 29 років).
- Кожен з користувачів промовляв різні команди в різних умовах: в тиші та при наявності шумів (музики, розмови та дощу), а також на різних відстанях: 20 см та 2 м.

Результати були обчислені за допомогою наступної функції

$$K = \frac{\sum_j^m \sum_i^n a_{ij} * b_I}{k} * 100\%$$

a_{ij} – j-та спроба і-го користувача;

b_I – ідентифікатор розпізнавання;

$b_I = \begin{bmatrix} 1, & \text{вдача} \\ 0, & \text{невдача} \end{bmatrix}$;

m – кількість спроб (по 8 спроб);

n – кількість користувачів (4 користувача);

k – кількість експериментів (всього 32).

Нижче наведені результати досліджень у таблиці 4.1.

Табл. 4.1 – Результати есперименталь них досліджень

Відстань / Шум	Тиша	При наявності шумів
20 см	96,88% (31 вдалих та 1 невдалих)	93,75% (30 вдалих та 2 невдалих)
2 м	84,38% (27 вдалих та 5 невдалих)	75% (24 вдалих та 8 невдалих)

В цілому це є непоганий результат, програмі вдається розпізнавати різні голоси та виконувати команди навіть при наявності інших шумів. В більшості випадків проблема виникає через акцент, адже програма розпізнає англійські слова з правильною вимовою. Намагаючись вимовляти слова відповідно до словника, програма значно краще розпізнає мову.

Висновки до розділу

В даному розділі було наведено опис вхідних даних та очікувані результати, а саме перелік вхідних фраз та очікуваний результат у вигляді виконання команди та відповіді.

Також проведено дослідження якості розробленої системи розпізнавання мови. Програма була протестована користувачами різної статі та віку (двоє жінок 20 та 30 років; двоє чоловіків 24 та 29 років), які промовляли команди в різних умовах: в тиші та при наявності шумів (музики, розмови та дощу), а також на різних відстанях: 20 см та 2 м. На

основі отриманих результатів було пораховано якість роботи програми у відсотках.

Глава 5. Функціонально-вартісний аналіз роботи.

5.1 Постановка задачі проектування

У даному розділі проводиться оцінка основних характеристик програмного продукту, призначеного для локального голосового управління персональним комп'ютером з вбудованою системою розпізнавання англійської мови. Програмний продукт призначено для використання на персональних комп'ютерах в якості персонального помічника. Інтерфейс користувача був розроблений за допомогою мови програмування Java у середовищі розробки NetBeans.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

5.2 Обґрунтування функцій та параметрів програмного продукту

Головна функціяF_0– розробка програмного продукту, який вирішує задачу розпізнавання англійської мови та виконує відповідні голосові команди, сказані людиною.

Виходячи з конкретної мети, можна виділити наступні основні функції програми:

F_1- вибір мови програмування: a) Java, b) Python.

F_2- вибір бібліотеки для розпізнавання: a) Sphynx, b) Kaldi

F_3- вибір середовища розробки: a) NetBeans, b)PyCharm

Виходячи з представлених варіантів будуємо морфологічну карту (рис.5.1).

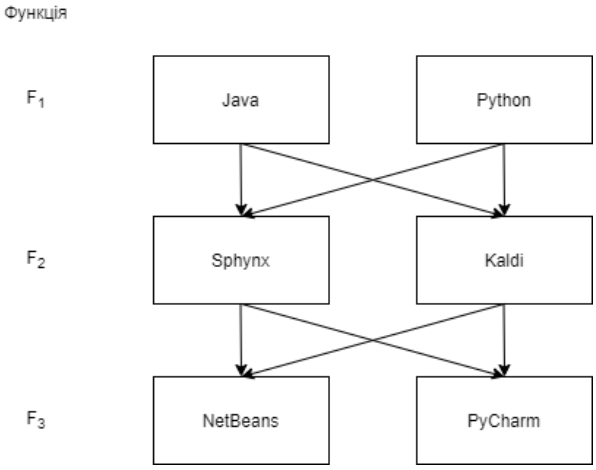


Рис. 5.1 Морфологічна карта.

Спираючись на карту була побудована позитивно-негативна матриця(табл.5.1)

Табл. 5.1 - Позитивно-негативна матриця

Основна функція	**Варіанти реалізації**	**Переваги**	**Недоліки**
F_1	А	Незалежність від платформи, багатопоточна обробка	Потребує багато пам’яті
	Б	Менше часу при написанні коду	Більший час на виконання операцій
F_2	А	Безкоштовність, може використовуватись як для Java, так і для Phyton	Складна в використання, вибаглива у мові розмітці
	Б	Безкоштовність, підтримує 35 мов, що доступні для Windows Speech Recognition	Підтримує тільки Windows Speech Recognition
F_3	А	Підтримує виділення синтаксичних конструкцій кольором, автодоповнення мовних конструкцій	Займає багато пам’яті

	Б	Підтримує редактор коду, зручна навігація	Повільний

Проаналізувавши позитивно-негативну матрицю отримуємо наступні варіанти реалізації програмного продукту:

1. $F_1\text{А} - F_2\text{А} - F_3\text{А}$
2. $F_1\text{А} - F_2\text{Б} - F_3\text{А}$

Обґрунтування функцій та параметрів програмного продукту

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче. Функція F3 залежить від двох параметрів X3 і X4(опис даних параметрів наведено в таблиці 5.2).

- X1 – відображає швидкодію операцій мови програмування залежно від обраної мови програмування;
- X2 – відображає об’єм пам’яті в оперативній пам’яті персонального комп’ютера, необхідний для збереження та обробки даних під час виконання програми;
- X3 – відображає час, який витрачається на дії;
- X4 – потенційний об’єм програмного коду.

Кількісні оцінки наведені нижче у таблиці 5.2.

Табл. 5.2 - Кількісні оцінки параметрів

Найменування параметру	Позначення параметру	Значення параметру		
		Мінімальне	Середнє	Максимальне
Швидкодія мови програмування (с)	X1	20	10	5
Об’єм пам’яті для збереження даних (мб)	X2	32	16	8
Час обробки даних (мс)	X3	800	550	100

Потенційний об’єм програмного коду(кількість рядків коду)	X4	1900	1500	900

За значеннями в таблиці побудуємо графічні характеристики параметрів

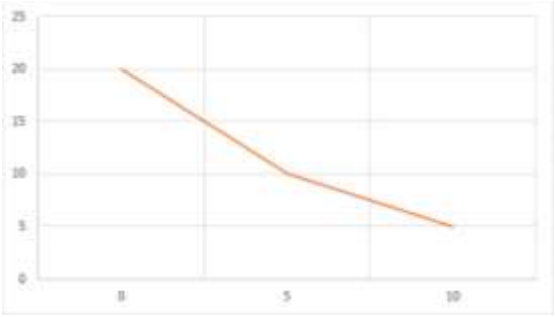


Рис. 5.2 Значення параметрів X1

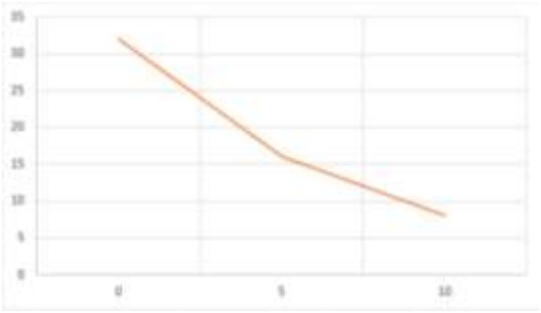


Рис. 5.3 Значення параметрів X2

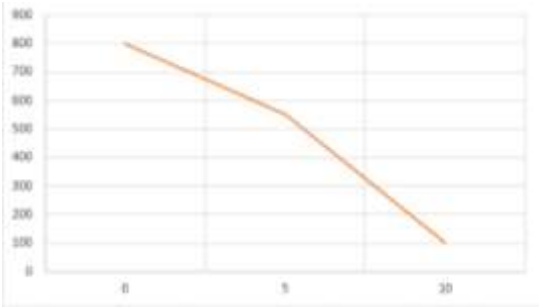


Рис. 5.4 Значення параметрів X3

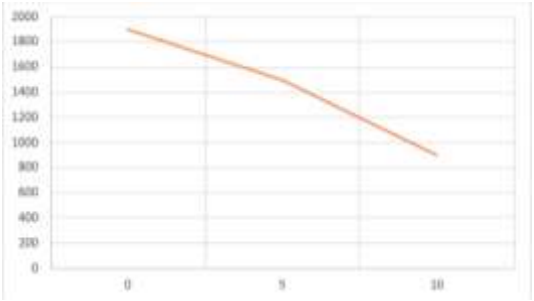


Рис. 5.5 Значення параметрів X4

Результати експертного ранжування наведені у таблиці 5.3.

Табл. 5.3. - Результати експертного ранжування

Параметр	Ранг параметру по оцінці експерта							Сума рангів, Ri	Відхилення Δi	Квадрат відхилення, (Δi)2
	1	2	3	4	5	6	7			
X1	1	1	2	2	1	1	1	9	-8.5	72.25
X2	2	2	1	1	2	3	2	13	-4.5	20.25
X3	3	4	5	4	5	2	3	26	8.5	72.25
X4	4	3	2	3	2	4	4	22	4.5	20.25
Разом	10	10	10	10	10	10	10	70	0	185

Визначемо коефіцієнт конкординації:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 * 185}{7^2(4^3 - 4)} = 0.755 > W_k = 0.67$$

Так як коефіцієнт конкординації більше нормативного, результати вважають достовірними.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 5.4.

Вважаємо, що експерт ставить ранг 4 найважливішому параметру, тоді відповідно 1 найменшому, як наведено у таблиці 5.4.

Табл. 5.4 - Попарне зрівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 та X2	<	>	>	>	<	<	>	>	1.5
X1 та X3	<	<	<	<	<	<	<	<	0.5
X1 та X4	<	<	<	<	<	<	<	<	0.5
X2 та X3	<	<	<	<	<	>	<	<	0.5
X2 та X4	<	<	<	<	<	<	<	<	0.5
X3 та X4	<	>	>	>	>	<	<	>	1.5

Як видно з таблиці 5.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Табл. 5.5 - Розрахунок вагомості параметрів

Параметри	Параметри xj				Перший крок		Другий крок		Третій крок	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i	$K_{\text{ві}}$	b_i	$K_{\text{ві}}$
X1	1	1,5	0,5	0,5	3,5	0,2186	12,25	0,209	44,875	0,21
X2	0,5	1	0,5	0,5	2,5	0,1562	9,25	0,156	34,125	0,16

X3	1,5	1,5	1	1,5	5,5	0,344	21,25	0,36	76,25	0,358
X4	1,5	1,5	0,5	1	4,5	0,2812	16,25	0,275	58	0,272
Загалом:					16	1	59	1	213,25	1

Обґрунтування функцій та параметрів програмного продукту наведено у таблиці 5.6.

Табл. 5.6 - Розрахунок вагомості параметрів

Основна функція	Параметри	Варіант реалізації	Абсолютне значення параметру	Бальна оцінка параметру	Коефіцієнт вагомості параметру	Коефіцієнт якості
F1	X1	А	5	10	0,21	2,1
F2	X2	А	10	8	0,16	1,28
		Б	20	4	0,16	0,64
F3	X3	А	100	5	0,358	1,79
F3	X4	А	1500	5	0,272	1,36

Обрахуємо коефіцієнти якості кожного з варіантів розробки:

$$K_{я1} = 2{,}1 + 1{,}28 + 1{,}79 + 1{,}36 = 6{,}53$$

$$K_{я2} = 2{,}1 + 0{,}64 + 1{,}79 + 1{,}36 = 5{,}89$$

Оскільки варіант 1 має найбільший коефіцієнт якості, він є найкращим.

5.3 Економічний аналіз варіантів розробки

Для оцінки трудомісткості розробки спочатку проведемо розрахунок трудомісткості. Усі варіанти мають наступні основні завдання:

1. Розробка акустичної моделі та системи розпізнавання англійської мови;
2. Розробка програмної оболонки.

Також кожний з варіантів має додаткове завдання, яке є реалізацією розгалужених варіантів розробки незалежного модуля.

3.1 Програма відповідає голосом.

3.2 Програма відповідає текстом.

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3. Проведемо розрахунок норм часу на розробку та програмування кожного із завдань.

Для завдання 1 (ступінь новизни А та група складності алгоритму 1, трудомісткість дорівнює: $T_Р$ = 90 людино-днів,$К_П$ = 1.7,$К_{СК}$ = 1,$К_{СТ}$ = 0.8, $К_М = 1, К_{СТ.М} = 1$. Тоді загальна трудомісткість програмування першого завдання дорівнює:$T_1 = 90 * 1{,}7 * 1 * 1 * 0{,}8 * 1 = 122{,}4$ людино − днів.

Для завдання 2 використовується алгоритм третьої групи складності, степінь новизни Б:

$$T_2 = 27 * 0{,}9 * 1 * 1 * 0{,}8 * 1 = 19{,}44 \text{ людино} - \text{днів}.$$

Для завдання 3.1 (ступінь новизни Б, складність 2)

$$T_{2.1} = 27 * 1{,}08 * 0{,}8 = 23{,}33 \text{ людино} - \text{днів}.$$

Для завдання 3.2 (ступінь новизни В, складність 3)

$$T_{2.1} = 12 * 0{,}6 * 0{,}6 = 4{,}32 \text{ людино} - \text{днів}.$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$T_I = (122{,}4 + 19{,}44 + 23{,}33) * 8 = 1321{,}36 \text{ людино} - \text{днів};$$

$$T_{II} = (122{,}4 + 19{,}44 + 4{,}32) * 8 = 1169{,}28 \text{ людино} - \text{днів}.$$

Більш високу трудомісткість має варіант І.

В розробці беруть участь два програмісти з окладом 10000 грн. Погодинна заробітна плата працівників складе:

$$С_ч = \frac{10000 + 10000}{2 * 21 * 8} = 59{,}5 \text{ грн}.$$

Тоді, розрахуємо заробітну плату для обох варіантів:

1. $С_{ЗП} = 59,5 * 1321,36 * 1,2 = 94345,104$ грн
2. $С_{ЗП} = 59,5 * 1169,28 * 1,2 = 83486,592$ грн

Відрахування на соціальне страхування(22%)(грн):

1. $С_{ВІД} = С_{ЗП} * 0,22 = 94345,104 * 0,22 = 20755,92$ грн
2. $С_{ВІД} = С_{ЗП} * 0,22 = 83486,592 * 0,22 = 18367,05$ грн

Далі розрахуємо витрати на оплату однієї машино-години. Враховуючи, що вона обслуговує одного спеціаліста з окладом 10000 з коефіцієнтом зайнятості 0,2 то для однієї машин отримаємо:

$С_{Г} = 12 * 10000 * 0,2 = 24000$ грн

Враховуючи додаткову заробітну плату:

$С_{ЗП} = 24000 * (1 + 0,2) = 28800$ грн

Відрахування на соціальне страхування 22%:

$С_{ВІД} = 28800 * 0,22 = 6336$ грн.

Розрахуємо амортизаційні підрахунки(амортизація 25%, вартість ЕОМ 25000 грн):

$С_{А} = К_{ТМ} * К_{А} * Ц_{ПР} = 1,15 * 0,25 * 25000 = 7187,5$ грн

Розрахуємо витрати на ремонт та профілактику:

$С_{Р} = К_{ТМ} * Ц_{ПР} * К_{Р} = 1,15 * 25000 * 0,05 = 1437,5$ грн

Розрахуємо ефективний годинний фонд часу ПК за рік

$Т_{ЕФ} = (365 - 104 - 11 - 16) * 8 * 0,9 = 1684,8$ год

Розрахуємо витрати на електроенергію

$С_{ЕЛ} = 1648,8 * 0,6 * 0,2 * 1,75 * 2 = 692,496$ грн

Накладні витрати рівні:

$С_{Н} = 25000 * 0,67 = 16750$ грн.

Отже експлуатаційні витрати(грн):

$$С_{ЕКС} = 28800 + 6336 + 7187,5 + 1437,5 + 692,496 + 16750 = 61203,49$$

Тоді собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М-Г} = \frac{61203,49}{1684,8} = 36,33$ грн/год

Тоді виплати на оплату машинного часу складають:

$С_{М1} = 36,33 * 1321,36 = 48005,01$ грн

$С_{М2} = 36,33 * 1169,28 = 42479,94$ грн

Накладні витрати відповідно

$С_{Н1} = 94345,104 * 0,67 = 63211,22$ грн

$С_{Н1} = 83486,592 * 0,67 = 55936,02$ грн

Розрахуємо повну вартість розробки за варіантами

$С_{ПП} = С_{ЗП} + С_{ВІД} + С_{М} + С_{Н}$

1. $С_{ПП} = 94345,104 + 20755,92 + 48005,01 + 63211,22 = 226317,254$
2. $С_{ПП} = 83486,592 + 18367,05 + 42479,94 + 55936,02 = 213269,602$

5.4 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня $К_{ТЕР_j} = {К_{К_j}}/{С_{Ф_j}}$

$К_{ТЕР1} = \frac{6,53}{226317,254} = 28,85 * 10^{-6}$; $К_{ТЕР1} = \frac{5,89}{213269,602} = 27,62 * 10^{-6}$.

Оскільки варіант 1 має більше коефіцієнт техніко-економічного рівня, перший варіант реалізації програми є більш ефективним.

Висновки

Таким чином в цій дипломній роботі було розроблено систему розпізнавання мови на основі апарату штучних нейронних мереж. Також були отримані наступні результати:

- Було досліджено історичне підґрунтя, від самого першого засобу для розпізнавання мови до сучасного світу з розумними системами. Розібрано класифікацію систем розпізнавання, а також проведено порівняльний аналіз існуючих платформ.
- Детально розглянуто структуру мовлення та архітектуру розпізнавання. Виділено та проаналізовано основні етапи роботи алгоритмів розпізнавання мови, а саме запис аудіо, вилучення ознак, робота декодера на основі акустичної, мовної моделі та складання граматичного словника. В кожному етапі наведено використовувані алгоритми, які описані математично.
- Було створено та описано програму реалізації розпізнавання мови, наведені основні складові програми та детально розписано роль кожної зі складових. Надано інструкцію по застосуванню.
- Експериментально перевірено роботу програми при різних умовах та з декількома користувачами.

Література

1. Micheli G., Ernst R., Wolf W. Readings in Hardware/Software Co-Design. 2002. P. 147–158. URL: https://doi.org/10.1016/B978-1-55860-702-6.X5000-5
2. Karan B., Mahto K., Sahu S. Intelligent Speech Signal Processing. 2019. P. 153–173. URL: https://doi.org/10.1016/B978-0-12-818130-0.00009-X
3. Jat D., Limbo A., Singh C. Voice Activity Detection. 2019. P. 101–111. URL: https://doi.org/10.1016/B978-0-12-818130-0.00006-4
4. Lai E. Practical Digital Signal Processing. 2003. P. 1–13. URL: https://doi.org/10.1016/B978-075065798-3/50001-1
5. Sherman W., Craig A. Understanding Virtual Reality. 2003. P. 75–112. URL: https://doi.org/10.1016/B978-155860353-0/50004-5
6. Sherman W., Craig A. Understanding Virtual Reality (Second Edition). 2018. P. 10-256. URL: https://doi.org/10.1016/B978-0-12-800965-9.00004-0
7. Gales M., Young S. The Application of Hidden Markov Models in Speech Recognition. Foundations and Trends Journal. 2008. Vol. 1, No. 3. P. 195–304. URL: https://mi.eng.cam.ac.uk/~mjfg/mjfg_NOW.pdf
8. Jendoubi S., Yaghlane B., Martin A. Belief Hidden Markov Model for speech recognition. 2013. P. 1-18.
9. Park J., Chebbah M., Jendoubi S., Martin A., Belief Functions: Theory and Applications. 2014. Vol. 8764. P. 284.
10. L. Rabiner, A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of IEEE. 1989. Vol. 77. P. 257-286.
11. Ронжин А. Л., Будков В. Ю. Система протоколирования дикторов на базе алгоритма определения речевой активности в многоканальном аудиопотоке. Речевые технологии, 2010. № 3. С. 98—102.

12. Хайкин С. Нейронные сети: полный курс. Изд. 2-е, Москва: Вильямс, 2006. С. 1104.
13. Иванов В.И., Тимофеев М.В. Идентификация голоса человека на основе мел-частотных кепстральных коэффициентов. Международная молодежная научно-техническая конференция. Владивосток, 2016. С. 240–243.
14. Sun Microsystems. Java™ Speech Grammar Format Specification, Version1.0, October26,1998
15. JavaTM Speech API Programmer's Guide, Version 1.0, 26 October 1998.
16. Gers F., Schraudolph N., Schmidhuber J. Learning Precise Timing with LSTM Recurrent Networks. Journal of Machine Learning Research. 2002. Vol. 3. P. 115–143.
17. Karpov A., Kipyatkova I., Ronzhin A. Very large vocabulary ASR for spoken russian with syntactic and morphemic analysis. Proc. Interspeech-2011, Florence, Italy. 2011. P. 3161—3164.
18. Andrew W. Robinson J. Forward-backward retraining of recurrent neural networks. 1995. P. 743–749.
19. Robinson A.J. An Application of Recurrent Nets to Phone Probability Estimation. IEEE Transactions on Neural Networks. 1994. Vol. 5. No. 2. P. 298–305.

Додаток А

1. Основний клас - точка входу в програму.

Завантаження конфігураційних файлів та ініціалізація розпізнавання мови:

```
Configuration configuration = new Configuration();

configuration.setAcousticModelPath("resource:/edu/cmu/sphinx/models/en-us/en-us");
    String PathDictionary = "resource:/Jarvis/cfg/commands.dict";
    configuration.setDictionaryPath(PathDictionary);

    String pathLanguageModel = "resource:/Jarvis/cfg/en-us.lm";
    configuration.setLanguageModelPath(pathLanguageModel);
    Thread t1 = new Thread(new JarvisMainView());
    t1.start();
    LiveSpeechRecognizer jarvisRecognizer = new LiveSpeechRecognizer(configuration);
    jarvisRecognizer.startRecognition(true);
```

2. Захоплення аудіопотоку з мікрофону :

```
import java.io.InputStream;
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.TargetDataLine;
```

```
public class AudioCapture {
    private static float sampleRate = 44100;
    private static int sampleSize = 8;

    private static final int CHANNELS = 1;
    private static final boolean SIGNED = true;
    private static final boolean BYTE_ORDER = true;

    private static TargetDataLine line;
    private static int dataCaptureSize;

    public static void SetAudioFormat(float rate, int bits) {
        sampleRate = rate;
        sampleSize = bits;
    }

    public static void Initialize() {

        AudioFormat inputFormat = new AudioFormat(sampleRate, sampleSize, CHANNELS, SIGNED, BYTE_ORDER);
        DataLine.Info lineInfo = new DataLine.Info(TargetDataLine.class, inputFormat);

        try {
            line = (TargetDataLine) AudioSystem.getLine(lineInfo);
            line.open(inputFormat);
            line.start();
```

```
            dataCaptureSize = (int) Math.pow(2,
Math.floor(Math.log(line.getBufferSize()/5) / Math.log(2)));

        } catch (LineUnavailableException e) {
            System.err.println(e);
        }
    }

    public static byte[] CaptureMicrophoneInput() {

        byte[] data = new byte[dataCaptureSize];
        line.read(data, 0, data.length);

        return data;
    }

    public static InputStream getStream() {
        byte[] data = new byte[dataCaptureSize];
        line.read(data, 0, data.length);
        AudioInputStream ais = new AudioInputStream(line);
        return ais;
    }
}
```

3. Цикл обробки результатів та виконання команд

```
       new ProcessBuilder("cmd", "/c", "cls").inheritIO().start().waitFor();

        SpeechResult result;
        Boolean acceptCommands = false;
```

```
Boolean goodbye = false;


while ((result = jarvisRecognizer.getResult()) != null) {
   if (goodbye) {
    System.exit(0);
   }
   System.out.println("\\" + result.getHypothesis() + "\\");
   String command = result.getHypothesis();
   String task = null;
   String browserName;
   Process p;
   //  String otherAI = null;
   if(command.equalsIgnoreCase("Could you help me, Jarvis"))
   {
      acceptCommands = true;
      goodbye = false;
      JarvisReplyController jrc = new JarvisReplyController();
      jrc.Speak(new String[]{"Greetings, master",
         "   It's nice to hear your voice",
         "   You remember my name, so i presume you need something",
         "   You are either watching the Marvel movies or you need my help
again."
         });
   }
   if(command.equalsIgnoreCase("Goodbye, Jarvis"))
   {
      acceptCommands = false;
```

```
            goodbye = true;
            JarvisReplyController jrc = new JarvisReplyController();
            jrc.Speak(new String[]{"So you are self-dependent now?",
                "We will see how soon you come back asking for my help",
                "Finally. You've already wasted too much of my time"});
        }

        if (acceptCommands == true) {
            if(command.equalsIgnoreCase("Connect Office RDP")) {
                p = Runtime.getRuntime().exec("cmdkey /generic:" + "PC-
YILO" +
                    " /user:" + "XPAND/YILO" +
                    " /pass:" + "123456789");
                p.destroy();

                JarvisReplyController jrc = new JarvisReplyController();
                jrc.Speak(new String[]{"I am now connecting to your work
PC"});

                Runtime.getRuntime().exec("mstsc /v: " + "PC-YILO" + " /f
/console");

                Thread.sleep(2*60*1000);

                Process p1 = Runtime.getRuntime().exec("cmdkey /delete:" +
"PC-YILO");
                p1.destroy();
            }
            if (command.equalsIgnoreCase("Run Opera")) {
```

```
            browserName = "opera";
            Desktop desktop = Desktop.getDesktop();
            desktop.browse(URI.create(browserName));
            JarvisReplyController jrc = new JarvisReplyController();
            jrc.Speak(new String[]{"Don't forget to clean your browsing history afterwards. We all know what's in there"});

        }
        if (command.equalsIgnoreCase("Run Chrome")) {
            browserName = "chrome";
            Desktop desktop = Desktop.getDesktop();
            desktop.browse(URI.create(browserName));
            JarvisReplyController jrc = new JarvisReplyController();
            jrc.Speak(new String[]{"Just a warning. This app consumes so much resources and it is not even AI"});
        }
        if (command.equalsIgnoreCase("Open Notepad")) {
            task = "notepad";
            JarvisReplyController jrc = new JarvisReplyController();
            jrc.Speak(new String[]{"Feel free to write down all your thoughts"});
        }
        if (command.equalsIgnoreCase("Close Chrome")) {
            task = "taskkill /F /IM Chrome.exe";
            JarvisReplyController jrc = new JarvisReplyController();
            jrc.Speak(new String[]{"Good call. Always liked opera more"});
        }
        if (command.equalsIgnoreCase("Close Opera")) {
            task = "taskkill /F /IM Opera.exe";
```

```
            JarvisReplyController jrc = new JarvisReplyController();
            jrc.Speak(new String[]{"The Opera is dead. Long live the
Chrome"});
        }
        if (command.equalsIgnoreCase("Close Notepad")) {
            task = "taskkill /IM notepad.exe";
            JarvisReplyController jrc = new JarvisReplyController();
            jrc.Speak(new String[]{"I hope you saved everything. Not that i
care."});
        }
        if (command.equalsIgnoreCase("Call Skype")) {
            try {
                Runtime.getRuntime().exec(new String[] {"rundll32",
"url.dll,FileProtocolHandler", "skype:"});
            }
            catch (IOException e)
            {
                System.err.println("Cannot open skype. " + e.getMessage());
            }
            JarvisReplyController jrc = new JarvisReplyController();
            jrc.Speak(new String[]{"Just select who to call."});
        }
        if (command.equalsIgnoreCase("Close Skype")) {
            task = "taskkill /IM SkypeApp.exe";
            JarvisReplyController jrc = new JarvisReplyController();
            jrc.Speak(new String[]{"The skype is closed. Not minimized, but
completely closed. Checkmate microsoft"});
        }
        if (command.equalsIgnoreCase("Run Command Shell")) {
```

```
                task = "cmd";
                JarvisReplyController jrc = new JarvisReplyController();
                jrc.Speak(new String[]{"You can pretend that you controll this computer now."});
            }
            if (command.equalsIgnoreCase("Do I need an umbrella?")) {
                Desktop desktop = Desktop.getDesktop();
                desktop.browse(URI.create("https://sinoptik.ua"));
                JarvisReplyController jrc = new JarvisReplyController();
                jrc.Speak(new String[]{"Could have as well just look into the window."});
            }
            if (command.equalsIgnoreCase("Open any film")){
                Desktop desktop = Desktop.getDesktop();
                JarvisReplyController jrc = new JarvisReplyController();
                jrc.Speak(new String[]{"Want to watch something. Pick it yourself."});
                desktop.browse(URI.create("https://www.netflix.com/"));
            }
            if (command.equalsIgnoreCase("Empty Trash")){
                Runtime.getRuntime().exec("PowerShell.exe -NoProfile -Command Clear-RecycleBin -Confirm:$false");
                JarvisReplyController jrc = new JarvisReplyController();
                jrc.Speak(new String[]{"So now you want me to clean after you? What's next?"});
            }
            if (command.equalsIgnoreCase("Check Mail")){
              Runtime.getRuntime().exec(new String[] {"rundll32", "url.dll,FileProtocolHandler","Outlook"});
```

```
            }
            if (command.equalsIgnoreCase("Open Word")){
              Runtime.getRuntime().exec(new String[] {"rundll32",
"url.dll,FileProtocolHandler","WinWord"});
            }
            if (command.equalsIgnoreCase("Open Excel")){
              Runtime.getRuntime().exec(new String[] {"rundll32",
"url.dll,FileProtocolHandler","Excel"});
            }
            if (command.equalsIgnoreCase("Close Excel")){
               task = "taskkill /IM Excel.exe";
               JarvisReplyController jrc = new JarvisReplyController();
               jrc.Speak(new String[]{"Tables, tables too many tables."});
            }
            if (command.equalsIgnoreCase("Close Word")){
               task = "taskkill /IM WinWord.exe";
               JarvisReplyController jrc = new JarvisReplyController();
               jrc.Speak(new String[]{"That i can do. Just make sure that you've
saved your diploma."});
            }
            if (command.equalsIgnoreCase("Close Mail")){
               task = "taskkill /IM Outlook.exe";
               JarvisReplyController jrc = new JarvisReplyController();
               jrc.Speak(new String[]{"Worday is over. No more mail."});
           }

            if(task != null) {
              Runtime.getRuntime().exec(task);
            }
```

```
            }
        }
    }
```

4. Синтез відоповіді команди з текстового масиву:

```
package Jarvis.controller;

import com.sun.speech.freetts.Voice;
import com.sun.speech.freetts.VoiceManager;
import java.util.Random;


public class JarvisTTSController {

   public void TextToSpeech(String [] txt) {

       System.setProperty("freetts.voices",
"com.sun.speech.freetts.en.us.cmu_us_kal.KevinVoiceDirectory");
       String voiceName = "kevin16";
       Integer i;
       VoiceManager voiceManager = VoiceManager.getInstance();
       Voice JarvisVoice = voiceManager.getVoice(voiceName);

        if (JarvisVoice == null) {
            System.err.println(
                "Something went wrong");
            System.exit(1);
        }
```

```
        Random r = new Random();
        i = r.nextInt(txt.length);
        JarvisVoice.allocate();
        JarvisVoice.speak(txt[i]);

        JarvisVoice.deallocate();
    }
}
```

5. Перевірка доступності мікрофону

```
  public static void main(String[] args) {
        Mixer.Info[] mixers = AudioSystem.getMixerInfo();
        List<Line.Info> availableLines = new ArrayList<Line.Info>();
        for (Mixer.Info mixerInfo : mixers){
            System.out.println("Found Mixer: " + mixerInfo);

            Mixer m = AudioSystem.getMixer(mixerInfo);

            Line.Info[] lines = m.getTargetLineInfo();

            for (Line.Info li : lines){
                System.out.println("Found target line: " + li);
                try {
                    m.open();
                    availableLines.add(li);
                } catch (LineUnavailableException e){
                    System.out.println("Line unavailable.");
                }
            }
        }
```

```
        System.out.println("Available lines: " + availableLines);
    }
```

6. Передача вхідних даних з мікрофону до функції розпізнавання мови

```
public JarvisLiveSpeechRecognizer(Configuration configuration) throws IOException
{
    super(configuration);
    microphone = speechSourceProvider.getMicrophone();
    context.getInstance(StreamDataSource.class)
        .setInputStream(microphone.getStream());
}

    public JarvisLiveSpeechRecognizer(Configuration configuration, TargetDataLine tdl) throws IOException, LineUnavailableException {
        super(configuration);
        microphone = new JarvisMicrophone(tdl);
                // speechSourceProvider.getMicrophone();
        context.getInstance(StreamDataSource.class)
        .setInputStream(microphone.getStream());
    }
    public void startRecognition(boolean clear) {
        recognizer.allocate();
        microphone.startRecording();
    }
```

```
    public void stopRecognition() {
        microphone.stopRecording();
        recognizer.deallocate();
    }
}
```

7. Функція розпізнавання та декодування вхідного потоку:

```
    public void newProperties(PropertySheet ps) throws PropertyException {
        decoder = (Decoder) ps.getComponent(PROP_DECODER);
        monitors = ps.getComponentList(PROP_MONITORS, Monitor.class);

        name = ps.getInstanceName();
    }


    public Result recognize(String referenceText) throws IllegalStateException {
        Result result = null;
        checkState(State.READY);
        try {
            setState(State.RECOGNIZING);
            result = decoder.decode(referenceText);
        } finally {
            setState(State.READY);
        }
        return result;
    }
```

```
public Result recognize() throws IllegalStateException {
    return recognize(null);
}

private void checkState(State desiredState) {
    if (currentState != desiredState) {
        throw new IllegalStateException("Expected state " + desiredState
                + " actual state " + currentState);
    }
}

private void setState(State newState) {
    currentState = newState;
    synchronized (stateListeners) {
        for (StateListener sl : stateListeners) {
            sl.statusChanged(currentState);
        }
    }
}


public void allocate() throws IllegalStateException {
    checkState(State.DEALLOCATED);
    setState(State.ALLOCATING);
    decoder.allocate();
    setState(State.ALLOCATED);
    setState(State.READY);
}
```

```
public void deallocate() throws IllegalStateException {
    checkState(State.READY);
    setState(State.DEALLOCATING);
    decoder.deallocate();
    setState(State.DEALLOCATED);
}
```

Додаток Б

Розпізнавання мови на основі апарату штучних нейронних мереж

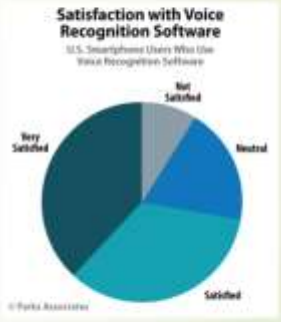
Ільченко Юлія Едуардівна
Керівник:
професор д.т.н Мухін Вадим Євгенович

План

1. Актуальність теми.
2. Історія та класифікація систем.
3. Архітектура розпізнавання.
А) Виділення ознак.
Б) Декодер.
4. Опис програми реалізації розпізнавання мови.
5. Робота програмного продукту "Jarvis".
6. Експериментальні дослідження.
7. Висновки.

Актуальність

Середньостатистична людина :
-говорить 150 слів за хвилину
-пише 40 слів за хвилину



Історія систем розпізнавання

- - Перший засіб для розпізнавання мови 1952 р. Bell Laboratories розробили систему "Audrey", яка розпізнавала вимовлені людиною цифри.
- - В 1962 році IBM випустила їх розроблену систему "Shoebox", яка розуміла 16 слів англійською.

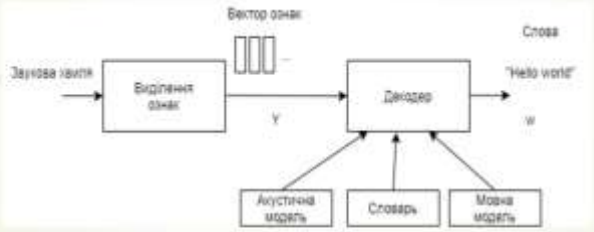
Історія систем розпізнавання

- - В 1997 році випустили перший у світі «безперервний розпізнавач мови» Dragon's NaturallySpeaking. Цей винахід був здатний розуміти 100 слів за хвилину, він все ще використовується сьогодні в оновленій формі і користується попитом у лікарів.

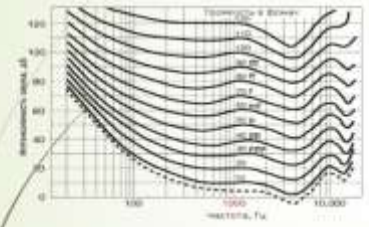
Історія систем розпізнавання

Класифікація систем

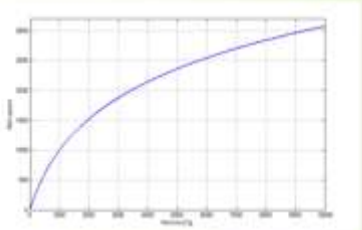
Архітектура розпізнавання



Виділення ознак



АЧХ вуха людини



Графік залежності мел-одиниць від частот

Мел-частотні кепстральні коефіцієнти (MFCC) - це функція, яка широко використовується в автоматичному розпізнаванні мови та динаміків.

Мел – це одиниця висоти звуку, заснована на сприйнятті цього звуку нашими органами слуху.

Декодер

Опис програми реалізації розпізнавання мови

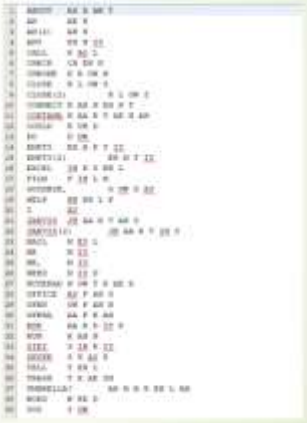
Мова програмування: Java

Середовище розробки: NetBeans

Опис програми реалізації розпізнавання мови

- Packages

1. Jarvis.cfg
   - ✓ Commands.dict
   - ✓ En-us.lm
2. Jarvis.controller
   - ✓ JarvisSpeechEngineController
   - ✓ JavaReplyController
   - ✓ JarvisTTSController



Запуск програми

- 1. Запускати "Jarvis.jar" (Executable Jar File) через командну строку.
- 2. Запускати "Jarvis_launcher.bat" (Windows Batch File).

Експериментальні дослідження

- Програма тестувалась користувачами різної статі та віку (двоє жінок 20 та 30 років; двоє чоловіків 24 та 29 років).
- Кожен з користувачів промовляв різні команди в різних умовах: в тиші та при наявності шумів (музики, розмови та дощу), а також на різних відстанях: 20 см та 2 м.

Відстань / Шум	Тиша	При наявності шумів
20 см	96,88% (31 вдалих та 1 невдалих)	93,75% (30 вдалих та 2 невдалих)
2 м	84,38% (27 вдалих та 5 невдалих)	75% (24 вдалих та 8 невдалих)

Висновки

- ✓ Досліджено історичне підґрунтя
- ✓ Розібрано класифікацію систем розпізнавання
- ✓ Розглянуто архітектуру розпізнавання
- ✓ Створено та описано програму розпізнавання мови
- ✓ Експериментально перевірено роботу програми при різних умовах та з декількома користувачами

JARVIS


ДЯКУЮ ЗА УВАГУ!
Questions?